Final Report
Flock of Birds System

Mark Krijgsman *Peter Curet*

Delft University of Technology

Delft
November 12, 2008

Preface

As a final assignment, students majoring in computer science at the Delft University of Technology are required to independently complete a project, often through an internship. The goal of the project is to build up experience, a process in which fellow students collaborate on an automation project and carry out its development process.

Traditionally, this project is scheduled for the last quarter of the third year of the bachelor, accounting for 15 ECTS points (420 hours). With permission from Ir. M. Sepers we completed our assignment in the first and second quarter of the year, during the months of September 2008 through March 2009.

The following thesis is the result of the work done by Mark Krijgsman and Peter Curet during this period, a product of collaboration between the Delft University of Technology, and the Leiden University Medical Center.

We would like to thank our project supervisors Dr. Ir. C. P. Botha and Ir. P. R. Krekel for their help and guidance through the entire process. Furthermore we would like to thank Dr. Ir. J. H. de Groot, and Drs. F. Steenbrink at the Leiden University Medical Center for their assistance and for the project assignment. Lastly we would like to thank Serkan Arikan and Lennart Bos who have provided us the groundwork for the project.

Peter Curet & Mark Krijgsman

Delft, April 6 2009

Summary

At the movement laboratory of the Medical University Centre Leiden, a sensor based tracking hardware called the Flock of Birds, is used for shoulder research. This system consists of several sensors that are attached to a patient's body. After attaching the sensors, a user measures several bony landmarks, after which movement measurements can be made.
The earliest software program used for the Flock of Birds system was a DOS based system, in which the user received no visual feedback throughout the measurement process. Because this was error-prone, a new software program was developed to provide real time movement and additional functionality not possible in a DOS based program: the Flock of Birds Visualizer.

Our project is focused on further improving the Flock of Birds Visualizer and to prepare it for actual use by the researchers. This has been accomplished through the following process: first, we researched the technologies needed to develop the program. This included, among others things, the examination of the Python programming language and the extra tools such as the VTK toolkit and wxPython. After this, we studied the existing program design and updated it wherever necessary, while maintaining the basic ideas of the design. These changes in the design were then implemented and, in the final phase, tested at the Medical University Centre Leiden.

The final product is able to communicate with the Flock of Birds system, allowing users to successfully make measurements on a patient. Any restrictions concerning the number of used sensors and/or bony landmarks have been removed, providing a completely user configurable and generic application. Preference and configuration files provide ways for the user to customize the application for their personal, preferred use. New functionality has been added such as the possibility to add new bony landmarks to the application and the Centre of Rotation movement calculation has been partially implemented.

There are still many possibilities in extending the functionality of the product.
The addition of a driver interface allows future developers to effortlessly integrate other tracking hardware, such as the OptoTrak, a system similar to the Flock of Birds system. Other possibilities include functionality to make comparisons between two different recordings and additional graphs in the Graphical User Interface in which statistical data can be read in real time.

Index

1 Introduction

As technology has advanced through the years, hardware such as the Flock Of Birds (FoB) system and the Optotrak have provided us with the possibility of accurately tracking movements and rotations in 3D space. So accurately in fact, that these technologies can be applied in fields of medical research.

The FoB hardware provides the possibility of precisely recording joint and shoulder movements in patients. Recordings can be analyzed, allowing researchers to use the data for shoulder movement examination. One example is the possibility of comparing the regression of the glenohumeral joint to the rotation of the scapula.
Through the digital visualization of recordings, medical specialists have an improved insight of joint movements in space, thus aiding them in diagnosing, preventing and treating shoulder disabilities.

The aim of our project is to develop an application to aid in the recording and visualization of joint movements. Presently, the department of orthopedics at the Medical University Centre Leiden (LUMC) is using a DOS based application to control the FoB hardware. This software provides no feedback during measurement and recording, increasing the chances of errors during the process. Furthermore, the resulting data gathered from this process is not visualized.

An advanced FoB specific software program exists, called Motion Monitor. Due to the cost and the extensive features of this application, the idea of creating an alternative FoB Visualizer was realized. This alternative would boast a more user-friendly interface, eliminating the excessive functionality of the Motion Monitor and focusing more on the shoulder movements.

The basis of this software was developed by fellow students of the Technical University of Delft (TU Delft) for their bachelor project. However, during their period working on the software, not all functionality was realized. We were assigned the task of completing the software and making it ready for actual use, working towards the goal of ultimately releasing it for non-commercial use as an open source application.

Below is a listing of the most important changes included in the final product:

- Overall cleanup of existing code, removing unnecessary methods, moving others and creating new classes to correspond with our new design.
- Removal of hardcoded elements wherever possible.
- Working link between the FoB hardware and software.
- Redesign of the GUI, in particular the Sensor and GH-Joint panels.
- Implemented more flexibility for the placement of bony landmarks.
- Implemented a separate menu for the creation of new bony landmarks.
- Translation of the CoR files from Matlab to Python.
- Implemented the possibility to disable the visualization model, bony landmark position indicators, or sensor position indicators.
- Implemented the possibility to load different models for visualization.
- Overhaul of the recording/playback options, improving functionality and making it more generic.
- Improved recording functionality by taking into account the different calibration matrices for sessions with different configuration.
- Implemented support for future OptoTrak drivers with the design and implementation of a driver interface.
- Addition of error- and consistency checking when navigating through the panels.
- Addition of timestamps to measurements.
- Addition of a separate menu for the detection of the COM ports listening to the FoB.
- Extended the possibility to configure the application by adding new sections in the configuration file and addition of error checking for all user input through the configuration file.
- Extensive documentation using Doxygen.

The following chapters further describe the process of development. In chapters 1 through 4, definitions and abbreviations used throughout this document will be further elaborated on, an insight into the workings of the FoB hardware will be provided, and the tools used during the project will be discussed. Following this introduction into the technical aspects of the project, we will continue by highlighting the development process starting in chapter 5, with the description of the project assignment.
Chapter 6 describes in detail the requirements for the application, and chapter 7 explains the design made, based on the requirements. We then continue with a discussion of the actual process in chapter 8, including a planning and a week-by-week description of activities. Chapter 9 lists any further extensions that can be made to the final product. We conclude the document with a conclusion in chapter 10, and an evaluation in chapter 11.

2 Definitions and abbreviations

Several definitions and abbreviations used throughout this report are clarified in the following table.

FoB	Flock of Birds
BL	See Bony Landmark
Bony Landmark	A bone point to be measured by the system
GH-Joint	Glenohumeral Joint
GUI	Graphical User Interface
LUMC	Medical University Centre Leiden
TU Delft	Technical University of Delft
VTK	Visualization toolkit
Rulebase	A set of rules used in decision-making
Comport	Serial data port used to communicate with the Flock of Birds hardware
CoR	Centre of rotation
OptoTrak	Optic tracking hardware
ECTS	European Credit Transfer and Accumulation System
Doxygen	Documentation system used in several programming languages
HTML	HyperText Markup Language, used to build web pages
Matlab	Numerical computing environment and programming language

3 Flock of birds

The FoB technology that is currently used in the project is a magnetic-transduction technique used for interactive computer graphics applications. The technology can be used for many purposes, in this case the measurement of anatomical parts for medical use.

The system consists of the major elements depicted in figure 1. These include a transmitter, transmitter driver circuit, sensor, and signal processing electronics.

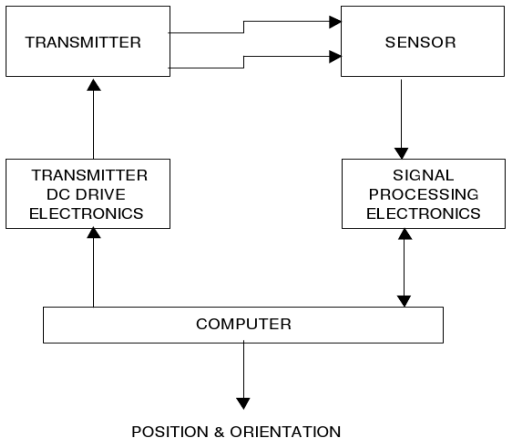

Figure 1: A graphical representation of the major elements of the flock of birds.[1]

The transmitter is located within a distance of approximately two and a half meters from the sensors. It consists of three individual antennae which are placed in such a way that they share a common centre. In such a way, a multiplicity of direct current magnetic fields are generated, and can be picked up by the sensors.

The position and orientation of one or several receiving antenna sensors can be measured by the flock with respect to the transmitting antenna, which is located at a fixed position. The mentioned receiving antenna sensors can be placed on the user's head, hand or body. Driving the transmitting antenna is a pulsing direct current (DC) signal.
The receiving sensor antennae measure the transmitted magnetic field pulse. The sensor consists of three axes of antenna that are sensitive to DC magnetic fields. Furthermore, it measures the earth's magnetic field. This magnetic noise is then removed from the sensor signal and in turn, improves the accuracy of the measurements. The sensor measures the position and orientation of the object to which it is attached, in our case a part of the human body.

Once measurements have been made, the signal output from the sensors is directed to the signal processing electronics. It processes the analogue sensor signals into a digital format that can be read by the computer. Next, an algorithm is used to compute the position and orientation of the sensor with respect to the transmitter and then the information is sent to the user's computer.
The resulting output contains the positions and orientations of the sensors with respect to the transmitters reference frame. The output of each sensor consists of three words which are the x, y and z positions. The user scales these words to any measuring unit, including inches and meters. The orientation of the transmitting sensor can be output as either nine words as the elements of a 3x3 orientation matrix, the three Euler angles azimuth, elevation and roll, or the four quaternion elements.

4 Tools

Throughout the project, several tools were used in order to complete the assignment. The following sections will give a outline of the tools used.

4.1 Python

Python is a programming language that mainly emphasizes four concepts.[2]

Quality
Python strives to keep a clear syntax, so that any code written in Python will be readable and easy to maintain. Python also focuses on easy extensibility, which is provided by a relatively small language core and an extensive standard library.

Productivity
Because the interpreter handles things such as type declarations, memory management and build procedures, (initial) development is easy so that productivity can be increased. The aforementioned point of clear syntax also increases productivity.
Furthermore, Python supports paradigms such as object oriented programming so that programmers can choose between several ways of designing their programs.

Portability
It is possible to run most Python programs under different operating systems, including Microsoft Windows, Macintosh OS X and Linux.

Integration
One of Python's uses is that of a "glue language". This means that Python can be used to connect several components made in different programming languages. For example, with Python it is possible to call C and C++ libraries, talk to Java classes and vice versa.

4.2 WxPython

One of the toolkits being used during the project is wxPython. This toolkit is used for the construction of the GUI and is implemented as a Python extension module that wraps around the GUI library wxWidgets. wxPython is open source and cross platform, currently supporting the 32-bit version of Windows, Macintosh OS X and most Unix systems.[3]

4.3 NumPy

NumPy is a package for scientific computing with Python, providing support for managing large data sets in arrays, rather than using the standard data structures such as tuples and lists. With NumPy, much less time and space is used to run programs that manipulate large sets of data.[4] Because the FoB data consists of matrices, NumPy is

essential to ensure that our program runs at reasonable speed and to provide functions needed to manipulate the arrays of data.

4.4 Visualization Toolkit

For the visualization of the recorded motion in the FoB application, the Visualization Toolkit (VTK) is used. It is an open source, object-oriented toolkit for the visualization of 3D data[5][6]. While it is implemented in C++, the toolkit can be used on other languages as well, including Python, which we have been using it for.
In the FoB application, VTK will be used for the visualization of the skeletal model. The model moves according to real time or recorded motion. Each bone is loaded into the application separately and added to a mapper. Then, actors referring each to a mapped bone are set to the right position. During the visualization, the angle and position of each actor and thus bone, is updated according to the real time or recorded data of the corresponding sensors.

4.5 Python Debugger

The Python Debugger (PDB) is a debug program which makes it possible to insert breakpoints, examine and change values at runtime and walk through the program step by step. Its behaviour is similar to the well known GNU Project Debugger (GDB).

4.6 Doxygen

To generate our documentation, Doxygen has been used. In order for Doxygen to successfully generate documentation, we have documented all classes, methods and variables. Extensive documentation is vital for the program after its release as open source application.

5 Project description

The assignment for our project is defined as follows:
Implement a link between the FoB hardware and software, re-implement the existing codebase in a more generic way, and add functionality for the CoR.

This definition describes our primary goals. Apart from these goals, we also have several optional goals, listed below:

- Possibility to disable the visualization model.
- Possibility to load in different models for visualization (especially for children, who might be intimidated by a skeleton visualization).
- Implement more flexibility for the placement of bony landmarks.
- Implement a more variable approach for making the landmarks.
- Implement support for the OptoTrak.
- Add timestamps to measurements.
- Implement a reset option for the visualization model to put the model back in a default position.

Because we continue work on an already developed system, our project is focused on improving and adding functionality to the already existing GUI, rather than making fundamental changes to the application.
The final product is a working software application for the FoB hardware which, next to the already existing GUI and visualization, includes the functionality to communicate with the FoB hardware and support for different kinds of measurements.

The project providers are:

- Dr. Ir. J. H. de Groot, Rehabilitation Medicine, LUMC
- Drs. F. Steenbrink, Orthopaedics, LUMC

The project supervisors are:

- Dr. Ir. C. P. Botha, Medical Visualization, TU Delft
- Ir. P. R. Krekel, Medical Visualization, TU Delft & Orthopaedics, LUMC

We had 420 hours (15 ECTS) available for our project, which has taken place in the practical halls at the Drebbelweg in Delft during development and at the LUMC during the testing phase. Regular meetings were scheduled with the project supervisors.

6 Requirement Analysis

During the design phase of the project, a requirement analysis was worked out as an aid during the development. The first and second subsections will respectively describe the current and the desired situation, followed by two subsections describing the functional and non-functional requirements.

6.1 Current situation

In the current situation an application has been developed, improving on the old DOS-based FoB application. This new application is not yet ready for practical use. Below, each of the four main tabs of the application are described first, followed by some general properties of the current system.

- For a measurement to be made, there are seven sensors that have to be attached to the patient. Three sensors per arm, namely on the lower arm, upper arm and acromion, and also one on the thorax. These sensors are visualized in a sensor panel and can be exchanged with each other if needed.
- When the sensors have been attached, the positions of the bony landmarks need to be calibrated, where bony landmarks are specific points on the bones that correspond with one or more of the attached sensors. This calibration is done with the use of a separate stylus sensor. Each bony landmark is measured five times by the stylus sensor, and with each measurement the stylus is tilted in a different angle. Then, these measurements are averaged to acquire a precise position. The GUI of this protocol has been designed in such a way that the user can see which measurements have to be made and whether they have been executed correctly. However, the actual functionality for making the measurements and checking for faulty measurements has not yet been implemented.
- A separate panel for the Glenohumeral Joint (GH-Joint) exists, which is meant to calculate the GH-Joints using either a regression or a movement method. Both of these methods are listed in this panel, but the functionality is yet to be implemented.
- The visualizer functionality has been partially implemented; a skeleton model is automatically loaded, and movement data to be visualized by the skeleton model can be imported.
- Many methods have been implemented based on hardcoded parameters, using hardcoded stylus sensor names and fixed lengths for bony landmark and sensor arrays.
- There is no link between the software and the hardware that is being used for the measurements.

6.2 The desired situation

We will now give an overview of the changes we will be making to create the new system. Each panel will be discussed, beginning with the Sensors panel.

- When starting up the application, the Sensors panel will automatically configure the current set of sensors two through nine. Feedback will be given concerning the position of the sensors.
  We have studied the possibility of having a separate advanced menu that is fully GUI based. This menu would allow the user to create, modify and delete sensors and corresponding bony landmarks. Bony landmarks can be appointed on a 3D-model. Furthermore, the rule base that checks for correct measurements can be adjusted in this menu as well.
  However, due to time constraints we have decided to implement a more hybrid approach where only a part of the menu is present in the GUI, and the rest is to be edited manually by an advanced user. This hybrid approach gives the user the ability to appoint and acquire bony landmarks coordinates on a 3D-model, but requires the user to manually edit the configuration files to add the name, description, rule, location and picture of a bony landmark.
- In panel two, the Bony Landmarks panel, the actual functionality of making and checking a measurement will be realised. The list of bony landmarks will also display new bony landmarks which may have been created in the Sensors panel.
- In the GH-Joint panel, the regression and movement method will be selectable. The movement method requires the patient to make circular movements with his or her arms. Following these measurements, the CoR can be estimated. The functionality for measuring these movements and to calculate the GH-Joint will be implemented as well.
- The Visualization panel will be extended to give support for different 3D-models to be loaded. There will also be an option to reset or disable the visualization model.
- The code will be adjusted to minimize hardcoded parameters and to support more variable options, such as extra flexibility concerning the placement and creation of bony landmarks. Timestamps will also be added to measurements.
- Specific drivers for communication between software and hardware of the FoB and OptoTrak system will be implemented.

6.3 Functional requirements

There are two actors who will make use of the system. Both of these actors are researchers, where the first actor acts as a standard user and the other as an advanced user. The standard actor should be able to perform the following actions:

- Attach sensors to the patient's body which will be checked for correct placement.
- Perform bony landmark measurements. Each bony landmark will be measured five times and coloured to indicate whether it is measured correct or not.

- Calculation of the GH-Joint(s), using either the regression or movement method. The regression method uses existing measurement for calculation, and the movement method prompts a menu guiding the user in making the new measurements.
- Record and save a new measurement in real time using the 3D-model in the Visualization panel.
- Playback recorded measurements in real time using the 3D-model in the Visualization panel.

The advanced actor should be able to perform all the actions of the standard actor, plus a set of additional actions listed below:

- Configure locations of the sensors on the body manually in the configuration file.
- Add, modify and delete new bony landmarks. When adding a bony landmark, the actor can select the location in a 3D-model, and manually edit the name, description, rule and picture in the configuration file.

6.4 Non-functional requirements

User interface and human actors
The user interface has to be intuitive and easy to use. The user has to be able to reach his or her goal in as few steps as possible. It should always be clear what the progress of the user is during measurements of the bony landmarks. Each bony landmark measurement is coloured to indicate whether a measurement is correct or not. In the Sensors panel, colours will also indicate whether the sensors have been placed correctly.

Documents
The following two documents are required for the system; the user manual, which can guide the user trough the application, and a documentation of all the functionality of the source code. The latter document can be consulted when there is need for maintenance or extensions to be made to the existing program. This is an important document, considering the fact that the software will be released as open source.

Hardware considerations
The application might be used for different sorts of hardware, namely the FoB system and OptoTrak system. This requires the software to be able to support both.

Performance characteristics
The system should be able to produce a fluid visualization of the model, both for recorded and real-time data.

Error handling and extreme conditions
Errors need to be handled with the use of user-friendly GUI messages, with information on the error and appropriate steps to be taken to correct the problem. Errors of any

kind may not crash the system. Extensive error checking needs to be implemented especially in situations where the user works with manually edited configuration files. Consistency should be maintained at any time. This means that users cannot navigate to the next menu when new sensors and bony landmarks have not yet been configured or when measurements have not been fully completed.

System modifications
The system has to have a design that is aimed on extendibility, so that extra functionality can easily be added to the system at a later stage if needed.

Physical environment
Since the system is influenced by electromagnetic fields, the environment needs to be free of distortion of any such kind. If the user does not take this aspect into account, measurements can be negatively influenced. Before using the system in a new environment, it needs to be calibrated to account for any present electromagnetic fields.

7 Design

During the design phase, adjustments were made to the existing design, both in the software architectural aspects as well as the interface aspects. In section 7.1 the packages are explained and in section 7.2 the classes are clarified. In section 7.3 the interface will be detailed.

7.1 Packages

The design of the software consists of several separated packages. Each package consists of classes serving one function as a whole. These packages are the main, the reader and the GUI, and are illustrated in the diagram in Figure 2.

Main

The main package contains the classes which form the core of the program, and control the entire process. This is done through a state machine design pattern, in which each state in the main package corresponds with a panel class in the GUI package. This allows the graphical elements to be separated from the functional elements.

Reader

The classes in the reader package handle the input and processing of information.
The driver classes control the FoB hardware by sending commands and receiving data. The DriverInterface class provides a template for hardware drivers, allowing classes from the main package to communicate with tracking devices in a uniform manner. The FobDriver and the planned Optotrak driver strictly inherit from this class. To simulate the hardware connection when the FoB hardware was not available to us, a separate driver was used which simulated the hardware using existing data files for visualization.

The PreferencesFileReader class performs the reading and writing of the user preferences from and to a file. These preferences include the paths to data files for the calibration, stylus, bony landmark and now the configuration file.
The ConfigFileReader reads the configuration set by the user from a designated configuration file. With this version of the FoB, expert users can now completely configure the system to their liking. Examples of the configurations users can adjust are: the active sensors, the sensor and bony landmark connections, and the 3D model utilized for the visualization.
Lastly, the RecordingProcessor class loads and writes serialized Recording objects representing a recording session to and from files.

GUI

This package handles the graphical elements of the program. Every state in the main package has a corresponding panel in the GUI package, representing a display of the state in which the program resides. Several dialog menus can also be found in the GUI

package. Additions to this version of the GUI are the ComportCheck menu and the addBonyLandmarksFrame.

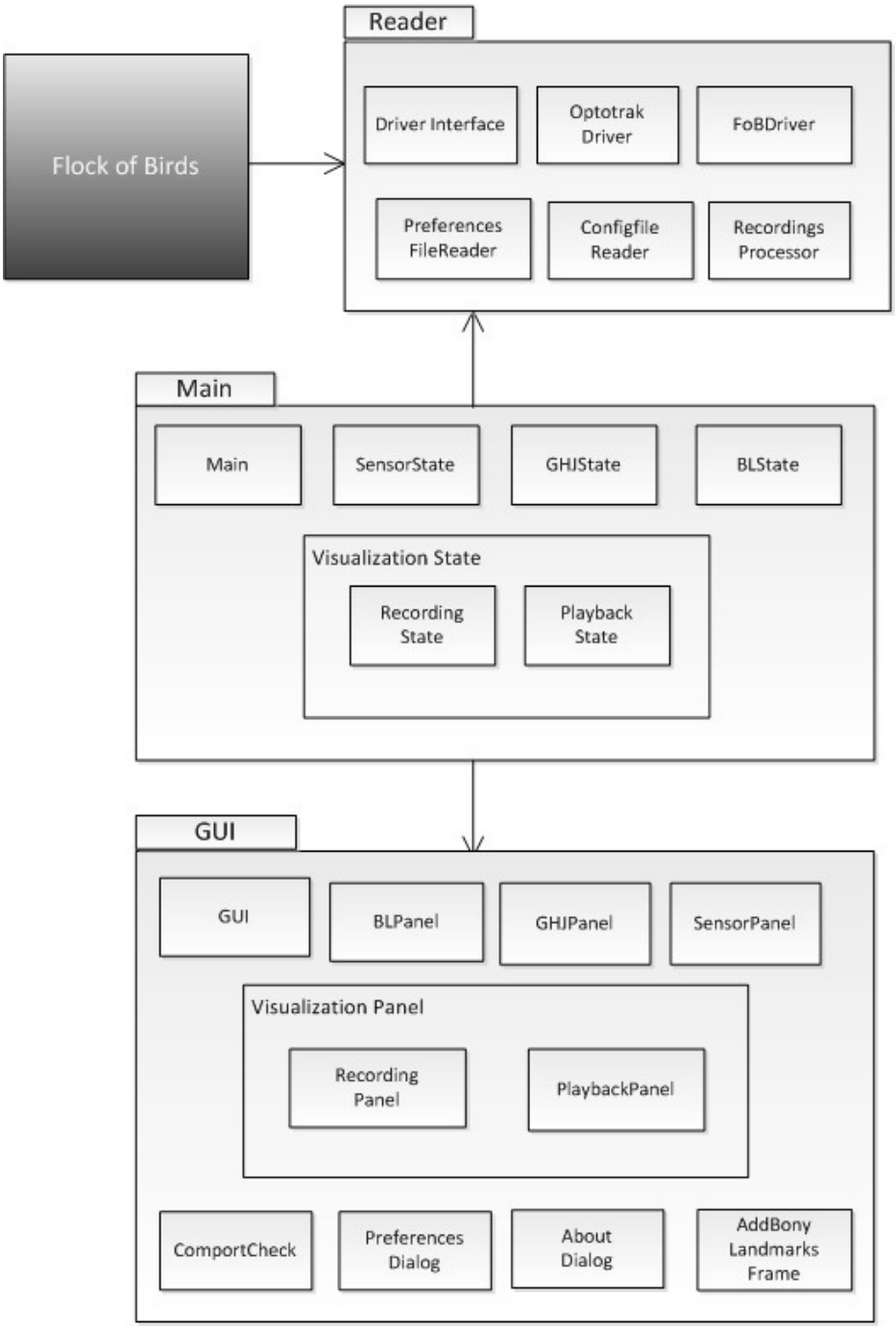

Figure 2: A package diagram of the FoB Visualizer.

7.2 Classes

In the following section we will elaborate on the design using the class diagram. In figure 3 a summarized version of the class diagram is shown.

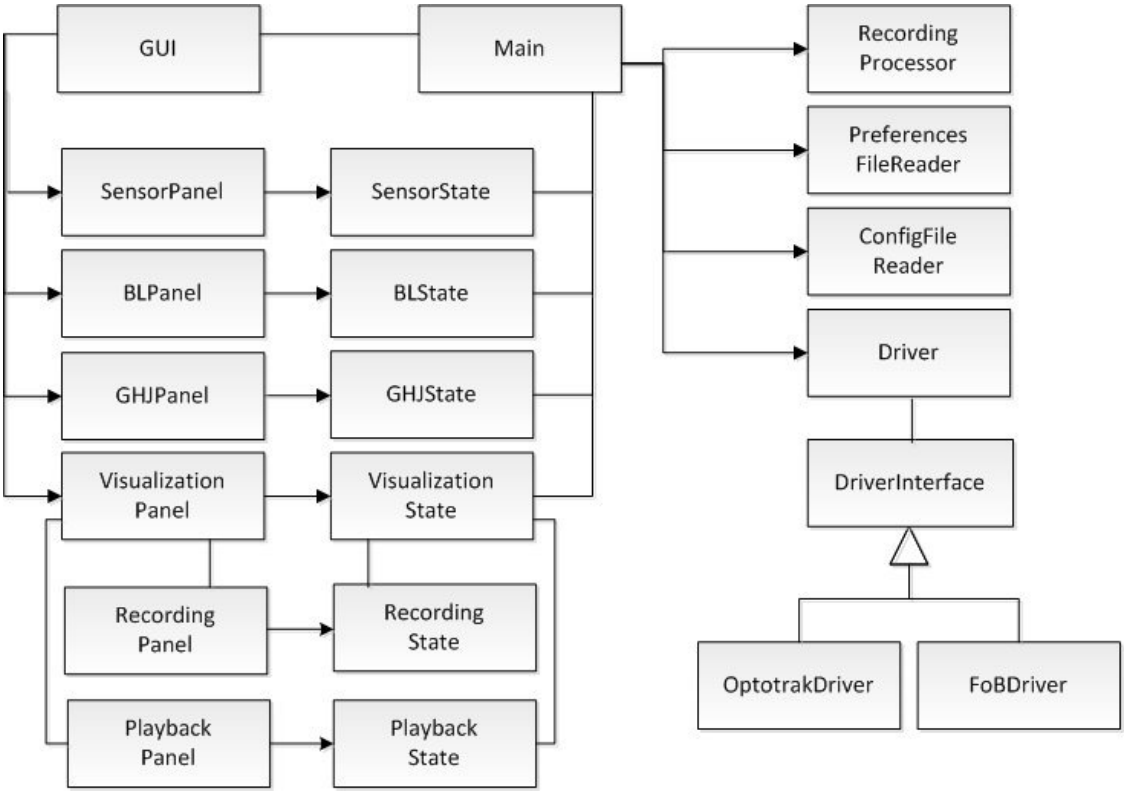

Figure 3: The class diagram.

During the project, one of our objectives was to make the application accessible for programmers. Using the tool Doxygen, a complete HTML based site has been generated with all classes, methods and parameters, along with complete English descriptions of each component.

7.3 GUI

Sensors Panel

In this first panel, the position of each sensor is indicated on a model. The corresponding name for each sensor is listed as well. After attaching the sensors to the patient, the user clicks “Next” and the position of each sensor is checked.

Bony Landmarks Panel
This panel will guide the user through the protocol of measuring the bony landmarks, shown in a list in the left side of the panel. The selected bony landmark is represented by an image and description in the middle of the panel. To measure a bony landmark, press the pedal button five times. After each click, the program will indicate whether a measurement was successful or not using green or orange respectively.
When the user clicks “Next”, the positions of all bony landmarks are compared to each other and again an indication is given whether the bony landmarks are correctly positioned.
The user also has the option to reset one or all bony landmarks using the “Reset selected” and “Reset all” buttons.

GH-Joint Panel
Here, the glenohumeral joints are calculated. There are two different methods available for calculation of these joints, regression and movement.
The regression method uses existing data from the previous panels to calculate the GH-Joints, whereas the movement method requires the patient to make circular movements which are recorded and then used to calculate the GH-Joints. A measurement for the movement method is started using the “Start/Stop Define” button for left and/or right.
The user also has the option to reset a measurement using the “Reset” button for left or right.

Visualization Panel
The visualization panel is divided into two subpanels: the recording panel and the playback panel.

- The recording panel shows a real-time visualization, where the user can make recordings using the recording button located in the bottom right of the panel.
- The user can playback earlier recorded measurements in the playback panel. Recordings made in the current session are automatically included in the recordings list.

Add bony landmarks Panel
As an aid for advanced users wanting to further configure the system, the add bony landmarks panel can be used to determine coordinates in the 3D model. These coordinates can then be used to add or reconfigure bony landmarks by specifying the coordinates as the position of a bony landmarks with respect to the model.

The panel is displayed as a frame separate from the main GUI, and visualizes a 3D model that can be rotated by being clicked on. Right-clicking will result in saving the click on position in a list. This list can then be copied to the clipboard, allowing the user to add the in the blpositions.ini file, or any file specified as the bony landmark position file in the preferences.

Check COM ports
As with the add bony landmarks panel, check COM ports is a panel separate from the GUI, and can be accessed through the menu of the main window.
Used when connecting the hardware to the visualizer, the check COM ports window displays a list of the COM ports, and the positions of the sensors connected to each COM port. This functionality aids users in checking whether the COM ports are connected correctly, and correspond with sensors that are being used.

Preferences dialog
In the preferences dialog, the user can change four files that determine the configuration of the application; the stylus file, calibration file, 3D bony landmark positions file and the configuration file.

8 Process

In this section we will detail the process and development during the project.
The project was divided into four phases, namely the orientation, design, implementation and testing phase. In the first subsection, the scheduling will be elaborated upon. In the second subsection, a complete week-by-week description of the processes will be detailed.

8.1 Planning

During the start of the project, a planning was made as estimation for the activities and the time spent during each phase. The entire course of the project was initially run from the beginning of September through the end of January spanning one semester of the year, as classified by the TU Delft. In the following figure the planned process is summarized in a Gantt-chart.

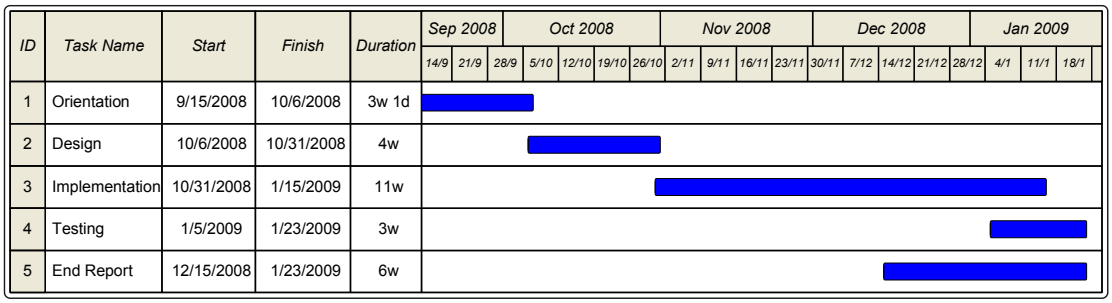

Figure 4: The planned development process displayed in a Gantt-chart.

During this period, a longer than expected implementation phase and problems during the testing phase caused several delays. Ultimately the project was worked on from the beginning of September to the mid of March. The actual time spent on each phase is illustrated in the following figure.

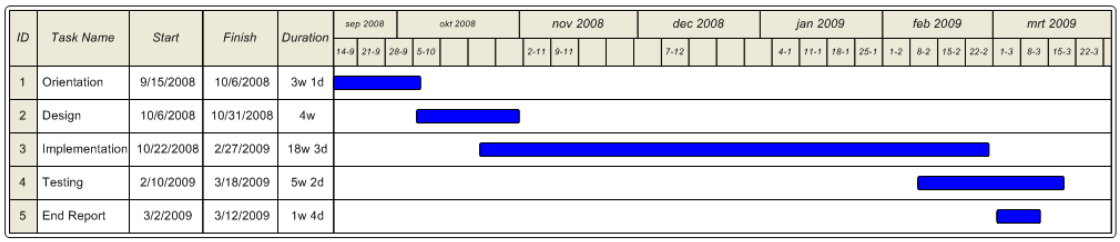

Figure 5: The actual development process displayed in a Gantt-chart.

8.2 Development Phases

As mentioned in the chapter introduction, the process involved four main phases.

Orientation

During the project we worked with several different technologies. To familiarize ourselves with these technologies, the orientation phase was essential in conducting research on the relevant areas of computer and medical science. The results of this research were described in an orientation report. Furthermore, to acquaint ourselves with the domain of the application, a visit was paid to the movement laboratory at the LUMC. In addition, several meetings with our supervisors helped us learn about the assignment and its development. Furthermore, the analysis of the previous project was an important factor in the orientation phase. The source code was examined carefully in order to fully comprehend the application.

Design

Once the orientation process was completed, the design of the FoB visualization application was analyzed for possible adjustments and new elements. As there already was an existing design for the application, our task was to find possible flaws in the design and to try and improve them. Familiarizing ourselves with the source and the architecture was still an important factor during this phase, as was analysing the primary assigned goals and incorporating them into the existing design.

Implementation

During the implementation phase, we realized the design modifications and additions shaped in the previous design process. During this phase the primary objectives were realized. Furthermore, due to the long time span of this phase, many secondary objectives were also implemented. Leading up to the end of the project, this phase was overlapped by the testing phase.

Testing

Once the majority of the newly designed elements had been implemented into the FoB visualization software, testing of the application became the primary objective.
With the help of pre-recorded data and a simulated driver class, much of the functionality had been tested at home. The actual connection between the hardware and software however, was done in the LUMC movement lab. Therefore several visits were made to the laboratory several times to connect the software to the FoB to assure the software was functioning correctly. In this phase several problems caused delays.

8.3 Development Process

We will follow with a week by week description of the development process.

Week 38 to 40

Starting the project with the orientation phase, the majority of the time was spent researching the hardware and the tools that would be used during the project.
A large portion of time also went into familiarizing ourselves with the existing code and its architecture. Furthermore, regular meetings with the supervisors were held to make sure we were working in the right direction. One of these meetings was with the supervisors at the LUMC. This meeting proofed especially fruitful in helping us set up our assignment.

Week 41 to 45

During these weeks the plan of approach and the design document have been written. Given that there already was an existing design, most of the time spent in this period was spent digging through the code. Several bug fixes were made in order to get the program running. With the program in operation, we could analyze the software to decide what would need to be changed in the current design. These results were incorporated into the design document. Furthermore during the analysis of the source code, the obvious hardcoded sections were fixed.

Week 46 and 47

During these weeks a to-do list was set up detailing each task and listed in the order of priority. Also, more orientation on the code was done. Part of the problem was figuring out how to complete the porting from block data to single sensor data which the supervisors had been working on. In this week more hardcoded elements were removed, one example is the pointer sensor which was always referred to as “fob0_2”.

Week 48

Since much of the code was not documented, each class was analyzed and commented. Documentation provided by the previous developers was used and translated. The idea behind this was that once the program was released as open source, each programmer would be able to understand what was happening.
Work on the button functionality was done, mainly for the next button of the first panel. Sensor checking using the rule base was activated.

Week 49

Work had been done on the rule base checking, there was a large bug in the rule checking caused by the porting of the data blocks to single data entities. Several changes to the interface were made to correspond with our ideas about the design, in particular that of the sensor panel.

Code cleanup proved to be a large task, determining what code was unused and moving several sections to appropriate classes. Furthermore, the fake fob driver could now be used for simulating real time sensor data by looping in the recording file.

Week 50
In this week, work on the recording and saving of real time data was done. This was completely overhauled and recoded from what was initially done.
Playback of these recordings was also implemented. In addition, the driver methods were rewritten, and an interface design was implemented, enforcing each of the driver classes inheriting from the main driver class to be implemented in a consistent way. Furthermore, code cleanup was done.

Week 51
Bugs and errors for the playback were worked on. Furthermore, proper functionality of the interface was implemented, with the disabling and enabling of buttons.

Week 52
The playback panel was further worked on, implementing the importing, renaming and deleting functionality. Additionally, work on the AddBonyLandmarkPanel started this week, a frame in which the 3D skeleton model can be clicked on to determine coordinate positions for the bony landmarks.

Week 3
Code and error cleanups were done. Another problem with the importing of recordings was solved, as the importing of a recording used bony landmark data of the current session. Several features were implemented to make sure the recordings were scaled and visualized according to the measurements of the corresponding sessions.

Week 4
Interface implemented for the GH-Joint Panel, providing the possibility for users to measure the centre of rotation to estimate GH joints. The related functionality however, is not yet implemented.

Week 5
During this period, the navigation through the menus was written, as the previous one proved to be faulty. Furthermore preferences can now be changed directly through the dialogs. Additional functionality for the AddBonyLandmarkPanel was implemented.
This week proved especially productive for the advancements we made concerning the accessibility of the code. The tool we came across, Doxygen, would be able to generate a complete HTML documentation of our code.

Week 6
Changes to the GH-Joint Panel were made once again. The additional functionality of toggling the 3D model in the visualization was implemented. Furthermore, bugs

concerning closing the application were solved. Also, to make our code compatible with the Doxygen tool, all the documentation was rewritten to be compatible with this tool, including method, parameter and return descriptions. The existing documentation was improved as well. In addition, a large code cleanup was once again done.
This week a start was made in translating the Centre of Rotation function from Matlab into Python. Most importantly, this was the first week in which we tested the software with the actual FoB hardware. Disappointingly, connecting the hardware and the software was not without problems.

Week 7
Several minor changes to the code were made. Additional testing was done at the LUMC. The problems however, prevailed. Further functionality was added to the configuration file and the corresponding reader class.

Week 8
Large steps were taken in making the program more generic this week. Extra bony landmarks and sensors can now be added through the configuration file without any errors.
Recording with these extra sensors and bony landmarks is now possible. Also, importing will not cause any problems and will use the configuration used when the import was recorded. Hardcoded coordinates of the sensor locations in the body picture located in the Sensor Panel were removed, and can now be configured through the configuration file. In addition timeStamp visualization for the Recording and Playback was implemented.
Furthermore, the existing FoB driver was partially rewritten to work with only a few sensors for testing on the mini-FoB. For this, two sensors were connected to the mini-FoB. After successfully connecting both sensors at the LUMC, a simple GUI was set up with two cones representing the sensor movements.

Week 9
Making use of the progress made on the mini-FoB, problems with the FobDriver were finally solved, allowing us to freely communicate with the FoB during the testing at the laboratory. Loading of the program was made more efficiently. Due to the connection of the USB-hubs through the COM ports, it was necessary to create a separate COM port-check class. This class allows users to see which COM port defined in the configuration file is connected to which sensor.

Week 10
The COM port check window is now integrated into the FoB software.
Working toward a generic application, configuration file reading functionality was added, allowing only active bony landmark (bony landmarks which are connected to the active sensors) to be loaded into the software. This was also implemented for the rulebase, and the GH-Joints. GH-Joints are now read through ConfigFileReader as well, and are created dynamically. Also, more generic GH-joint regression calculation has been made

configurable through the configuration file. In addition, an optional functionality has been added allowing users to specify the 3D model of their choice in the configuration file.

Week 11
Visualization was changed so that only bones associated with active bony landmarks would be visualized. This was an important step in making the program truly configurable, allowing expert users completely configure the system, making it usable for other functionality besides shoulder movements. Some error handling concerning the reading of configuration and preferences data has been implemented, giving users error dialogues. This is mainly a net for expert users who are configuring the system and might have made adjustments that are causing inconsistencies.

Week 12
Further testing was done. While the data of the simulation was visualized correctly, the new data that was being read from the FoB hardware was displayed improperly. To be able to do testing at home, the code was modified to export current FoB hardware data, to be used in the simulation. This allowed us to look for the error at home. Our suspicion when we found that one of the calibration matrices was not transposed, causing a faulty visualization. This led us to conclude that the old simulation data that we were given beforehand was somehow already modified.

Week 13-14
A meeting was held to evaluate the project process and to decide how we would conclude the project.

Week 15
Testing was done at the LUMC, and many changes were made this week. A few optional functionalities such as configurable 3D smoothing and resetting the camera were implemented. Furthermore the AddBonyLandmark was optimized. Beforehand, the point coordinates that were returned seemed incorrect. By using alternate VTK function for picking points, we were able to achieve far more accurate coordinates. Work on the final report was done, code was cleaned up and the doxygen documentation was further updated. We would like the emphasize the work that was done this week on entirely translating the CoR method from Matlab to Python. However, no testing was done to validate the correctness of the CoR method in the working environment. Another functionality was added to the program to check whether the hardware is connected or not. Whenever the FoB hardware is not connected, the visualize will only allow the user to import and play back older recordings. Furthermore, an extra dialog was programmed to be displayed whenever a COM port is not connected.

Week 16
Last testing was done to make sure all loose were tied up. Some final bug fixes such as navigational errors and visualization problems were corrected. Testing with

bonylandmark subsets was done to make sure all problems were solved. Manual was rewritten, and the configuration file was commented to further improve the documentation.

Week 17
A navigational error was fixed, final doxygen documentation was generated and uploaded, dialogs added. Config file was further commented.

9 Further extensions

Because our project concerned the improving of an already existing application, we did not have to build it from scratch. This made it possible for us to implement lots of the desired extensions and improvements. However, there are still extensions that are not included in the final product.

Different models
The possibility to load different models is desired for a select group of patients such as children. A more child friendly model could be loaded to comfort the child so that the measurement process is less troublesome.

Addition of drivers to control the OptoTrak
Apart from the Flock of Birds system, the LUMC has another similar measurement system, the OptoTrak. This system makes use of infrared light instead of sensors, but the basic principles remain the same as in the Flock of Birds system. Specific driver support has already been implemented in the final version. The only thing needed to support the OptoTrak is an additional driver version able to control the OptoTrak hardware.

Completion of the CoR method.
In the final version, the translation of the CoR Matlab files has already been made to Python. However, these Python files need to be verified for correctness and need to be tested by passing the correct parameters to each method.

Comparison of recordings
A desired extension is the ability to compare two recordings, where the application should be able to indicate differences between the patient's movements in both recordings. These differences could for example be the angles at which a patient can rotate or lift its arms. These comparisons should enable the user to conclude whether the situation has improved since treatment has started, or whether the treatment method needs to be altered.

Plots
A lot of data can be represented using plots. In these plots, the user can read numerical data to aid him in understanding the real time movements that are being visualized. These plots can also be examined at a later point in time to determine the treatment needed for the patient.

Optimalization of framerate
In the current situation, the application has a relatively low framerate due to the amount of calculations and visualization needed for a single frame. This has already

been improved by implementing options to disable the visualization model. However, this can be further improved by implementing functionality for a raw data function which only records the raw sensor data as produced by the FoB system. The raw data then has to be post-processed after recording.
During our testing phase we only had our laptops available with limited processor power, we expect better framerates when using higher end PC's for the real time visualization.

10 Conclusion

The assignment for our project has been defined as follows:
Implement a link between the FoB hardware and software, re-implement the existing codebase in a more generic way, and add functionality for the CoR.

To accomplish this, we used the already available FoB Visualizer as a starting point. Getting into a new project with an already exisiting codebase proved difficult, as we needed quite some time to get familiar with the system as a whole. After this orientation phase we made some necessary changes and bugfixes to get the application up and running.
After we gained enough knowledge of the system, we had a meeting at the LUMC to discuss the primary and secondary goals with the project supervisors. This meeting was very important, as it was the starting point where we based our design decisions on. For this design, we renamed several components, moved and removed methods between classes and added new ones such as the DriverInterface. This design proved to be successful during our development and has not been deviated from during actual implementation.

In the design phase, we reimplemented the functionality for all the panels to be more generic and also adding more functionality. During this reimplementation, we built on the ideas we were represented by the old hardcoded functions and reimplemented those in a different way. Changes to the GUI were also made to reflect our new design, including the addition of a menu to add new bony landmarks.

In week 20, we optimistically started off with our testing at the LUMC. However, the first three weeks proved unsuccessful as we could not get the FoB system running. Following this, when we managed to get the FoB system running, we had more troubles than expected to get a proper visualization due to the large amount of changes with respect to the old data files we had been simulating with up until that point.
Other factors also influenced the extra time we had to put in, such as the implementation of the COM port check class which could only be done at the LUMC. Altogether, this resulted in another five weeks of testing at the LUMC.
Because of the amount of unforeseen testing needed, we faced a large delay with respect to our initial planning.

The result of our extensive testing has not been without benefit, as the final product contains each primary and secondary goal, with the exception of a specific OptoTrak driver and a fully functional CoR method. However, these goals have been partially realized with the use of a DriverInterface for the OptoTrak and a completely translated CoR Python class.

11 Evaluation

In this section we will briefly evaluate the process of working on the FoB Visualizer.

As we had worked together previously in groups and on practical assignments, we decided to work together on the final bachelor project.
After orientating ourselves on a variety of projects, we met with Charl Botha who sparked both our interests: to work on an application to be used in the area of medicine.

Subsequently, we agreed with our supervisors and the bachelor project coordinator to start our project in September of 2008. During this start, we made an initial plan of approach, adjusted the design of the existing software, made a requirement analysis, and created a planning. This planning estimated the project to be finished by the end of January 2009. After what seemed a successful implementation phase, we started on the testing phase in which we came to the conclusion that we needed more time to work on the software and hardware connection than initially thought. At the end, we completed the project in the month of April, with the final presentation being in May.

Our working process involved a balanced workload distribution. When meeting together, we carefully evaluated what work had been completed, and what still needed to be done. These meetings would take place at the EWI faculty, at the LUMC or at our homes. Besides these regular meetings with each other, we maintained constant contact through e-mail and online messaging. Furthermore, we had meetings with our supervisors to evaluate and track our progress, to assure we stayed on the right track.

In addition to speaking with our supervisors, we would occasionally contact our project providers at the LUMC for help concerning the FoB hardware and certain portions of the code. Furthermore, during the testing phase which we would spend in the laboratory at the LUMC, they provided help whenever necessary.

Overall, the collaboration process was effortlessly without any problems. Besides the occasional scheduling conflicts, we managed to work together actively, to eventually deliver a working product. As much of the tasks were divided, Mark worked largely on the functionality for adding bony landmarks, GUI editing, COM port checking, documentation, and translating the CoR function. Peter worked on making the code more generic, proper visualization, and importing and exporting recordings and the rule base checking.
However, because the functionalities are intertwined, it was necessary to work in all sections of the code. This can be seen as an advantage, as we always understood each section of the code. Most of the software-hardware link functionality was written in cooperation during the time at the LUMC lab.

Because of the long testing period, we had a lot of time left in between testing days to implement extra functionality. This enabled us to deliver a complete working project,

with almost all optional goals and the inclusion of many error checking and user safety nets. This would not have been possible, would there have been a strict deadline.

12 Sources

[1] *Flock of Birds - Six Degrees of Freedom Measurement Device - Technical Description of DC Magnetic Trackers*
Ascension Technology Corporation, Burlington, VT
http://www.5dt.com/downloads/3rdparty/fobtechnicalspec.pdf
Retrieved on 06-10-08

[2] *Programming Python: Object-Oriented Scripting (2001)*
Mark Lutz
http://books.google.nl/books?id=37_AJDlEytEC&printsec=frontcover&hl=en
Retrieved on 28-09-08

[3] *What is wxPython*
http://wxpython.org/what.php
Retrieved on 28-09-08

[4] *Numerical Python (2001)*
David Ascher, Paul F. Dubois, Konrad Hinsen, Jim Hugunin, Travis Oliphant
http://www-int.stsci.edu/~jinger/ad5/web/ad4/irafx/docs/numpy.pdf
Retrieved on 28-09-08

[5] *What Is VTK?*
http://www.vtk.org/what-is-vtk.php
Retrieved on 06-10-08

[6] *The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization*
William J. Schroeder, Kenneth M. Martin, William E. Lorensen, GE Corporate Research & Development
http://www.vtk.org/pdf/dioot.pdf
Retrieved on 28-09-08

Appendix A: Orientation Report Document

1 Preface

The project we will be working on is provided by staff of the movement laboratory at the LUMC, which is the University Medical Centre in Leiden, in cooperation with the Technical University of Delft (TUD).
During the project we will be working with different technologies. To familiarize ourselves with these technologies, we have conducted research in the relevant areas of computer and medical science. The results of this research will be described in more detail in the following sections.

The work we will be doing is in succession of a previous project, also aimed at improving the Flock of Birds (FoB) application. The Flock of Birds system is a magnetic-transduction technique used for measuring motion, applicable to medical research.
During the previous project, a GUI was made to assist the user in making measurements with the FoB system. A 3D visualization of recorded measurements has also been made possible. In the current situation, the GUI and 3D visualization are not in use yet. Our aim is to improve on the current application by implementing the missing functions essential to the application.

We will continue the following report with a further elaboration of the project assignment we have been given. Subsequently, we will look at the available hardware which we will be working with. In the following section we will look into the software needed to further develop the application. In the last section we will look into the medical portion of the assignment, in order to fully understand the biological aspects in which the project domain is situated.

2 Project Description

During this project, we have different goals, some of which are primary and others which are optional. These optional goals will be evaluated after the primary goals have been completed. The primary goals are listed below.

- Implement a real-time link between hardware and software
- Make the system more generic
- Implement the Centre of Rotation

The most important goal is to implement a link between the hardware and software of the FoB system, making the system ready for practical use. After this has been realized, parts of the existing code will be re-implemented to make the system more generic. This means that sensors can be put on the patient in any order on any body part, allowing the medical specialist to choose which part of the body movements will be measured. In the current application, only shoulder movement measurements are supported. The third goal is to implement functionality that enables the FoB system to make measurements for the shoulder's Centre of Rotation (COR).

3 Hardware

When measuring shoulder movements, the current hardware used in the application is the Flock of Birds technology. Furthermore, the laboratory has also made the OptoTrak hardware available to us for use with the application. We will look into both tracking system technologies and further elaborate the possibilities of their use.

3.1 Flock of Birds

The Flock of Birds technology that is currently used in the project is a magnetic-transduction technique used for interactive computer graphics applications. The technology can be used for many purposes, in this case the measurement of anatomical parts for medical use.

The system consists of the major elements depicted in figure 1. These include a transmitter, transmitter driver circuit, sensor, and signal processing electronics.

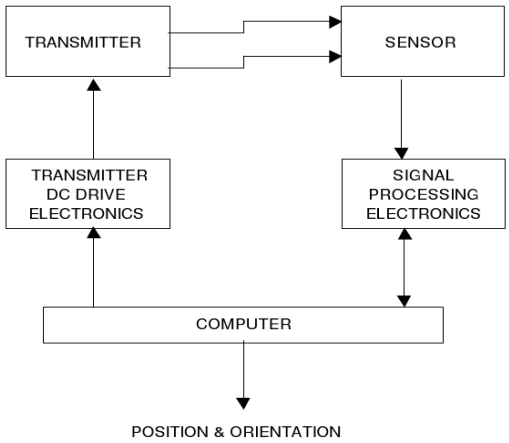

Figure 1: A graphical representation of the major elements of the flock of birds[1]

The transmitter is located within a distance of approximately two and a half meters from the sensors. It consists of three individual antennae which are placed in such a way that they share a common centre. In such a way, a multiplicity of direct current magnetic fields are generated, and can be picked up by the sensors.

The position and orientation of one or several receiving antenna sensors can be measured by the flock with respect to the transmitting antenna, which is located at a fixed position. The mentioned receiving antenna sensors can be placed on the user's head, hand or body. Driving the transmitting antenna is a pulsing direct current (DC) signal.
The receiving sensor antennae measure the transmitted magnetic field pulse. The sensor consists of three axes of antenna that are sensitive to DC magnetic fields. Furthermore, it measures the earth's magnetic field. This magnetic noise is then

removed from the sensor signal and in turn, improves the accuracy of the measurements. The sensor measures the position and orientation of the object to which it is attached, in our case a part of the human body.

Once measurements have been made, the signal output from the sensors is directed to the signal processing electronics. It processes the analogue sensor signals into a digital format that can be read by the computer. Next, an algorithm is used to compute the position and orientation of the sensor with respect to the transmitter and then the information is sent to the user's computer.
The resulting output contains the positions and orientations of the sensors with respect to the transmitters reference frame. The output of each sensor consists of three words which are the x, y and z positions. The user scales these words to any measuring unit, including inches and meters. The orientation of the transmitting sensor can be output as either nine words as the elements of a 3x3 orientation matrix, the three Euler angles: azimuth, elevation and roll, or the four quaternion elements.

3.2 OptoTrak

Next to improving the software for the FoB system, one of our optional goals is to make the software more generic. This means that functionality could be implemented so that the application can handle both the FoB and OptoTrak hardware effortlessly.
The OptoTrak is a system capable of performing measurements similar to those of the FoB system. However, the way both systems obtain these measurements are quite different.

The OptoTrak makes use of infrared sensors, which are being tracked by several cameras. These cameras are mounted onto a unit called a position sensor, as pictured below in figure 2.

Figure 2: OptoTrak position sensor with standard[2]

The infrared sensors, called markers, are attached to the subject, where three or more markers are needed to determine the 3D orientation of the body[3]. Because the OptoTrak is an optical system, the markers have to be in view of the position sensor at

all time. If the position sensor loses track of a marker, it is automatically indentified by the system when they return to view.

In order to further the precision of the movement measurements, the OptoTrak system can be used in combination with the FoB hardware. The results of both systems cancel out each others error, and thus resulting in highly accurate measurements.

4 Software

In order to build on the existing project, it will be important to understand the concepts of the software which will be used. In the following sections we will look into several software tools needed in the development process.

4.1 Python

Python is a programming language that mainly emphasizes four concepts.[4]

- *Quality:*
  Python strives to keep a clear syntax, so that any code written in Python will be readable and easy to maintain. Python also focuses on easy extensibility, which is provided by a relatively small language core and an extensive standard library.
- *Productivity:*
  Because the interpreter handles things such as type declarations, memory management and build procedures, (initial) development is easy so that productivity can be increased. The aforementioned point of clear syntax also increases productivity.
  Furthermore, Python supports paradigms such as object oriented programming so that programmers can choose between several ways of designing their programs.
- *Portability:*
  It is possible to run most Python programs under different operating systems, including Microsoft Windows, Macintosh OS X and Linux.
- *Integration:*
  One of Python's uses is that of a "glue language". This means that Python can be used to connect several components made in different programming languages. For example, with Python it is possible to call C and C++ libraries, talk to Java classes and vice versa.

4.2 Integrated Development Environment

The Integrated Development Environment (IDE) we will be using to edit and compile our code with is Eclipse. An extension for Eclipse exists, called Pydev, which provides a syntax highlighter, code completion and syntax analysis. [5]

4.3 WxPython

One of the toolkits being used during the project is wxPython. This toolkit is used for the construction of the GUI and is implemented as a Python extension module that wraps around the GUI library wxWidgets. wxPython is open source and cross platform, currently supporting the 32-bit version of Windows, Macintosh OS X and most Unix systems[6].

4.4 NumPy

NumPy is a package for scientific computing with Python, providing support for managing large data sets in arrays, rather than using the standard data structures such as tuples and lists. With NumPy, much less time and space is used to run programs that manipulate large sets of data [7]. Because the FoB data consists of matrices, NumPy is essential to ensure that our programs runs at reasonable speed and to provide functions needed to manipulate the arrays of data.

4.5 Visualization Toolkit

For the visualization of the recorded motion in the FoB application, the Visualization Toolkit (VTK) is used. It is an open source, object-oriented toolkit for the visualization of 3D data[2]. While it is implemented in C++, the toolkit can be used on other languages as well, including Python, which we will be using.
VTK has been created with a couple of common conventions to assure the consistency of the code. One example is the method names which have all been fully spelled out. The result makes it much easier for developers to remember methods. Furthermore, all member variables are either private or protected and can only be accessed from other classes through Set() and Get() methods.
Streamlines, ISO-contouring, glyphing, clipping, and cutting are part of the wide range of visualization techniques available through the toolkit. The API of VTK includes objects such as lights, cameras and actors, which can all be displayed within a render window. This window creates a viewport for each of its renderers and in turn gives them control to set up a camera and lights. Following this, each actor is rendered. All actors have properties and each are independent of each other.
Each object in the API has a broad range of functionalities. Examples are the parameters of lights which can be adjusted to have colour, and the possibility to adjust the parameters so that infinite light or spotlights can be simulated. Actor objects have a matrix transformation to indicate its current positions and scale in the world coordinates. Furthermore, it has properties such as ambient, diffuse, and specular colours, wire-frame versus surface rendering, and front- or back-face culling.
This graphics model contains the essential properties of a standard 3D graphics system. It is based on movie-making, and graphical user interface windows applications. Figure 3 gives a basic explanation of the VTK graphic model and its objects[9].

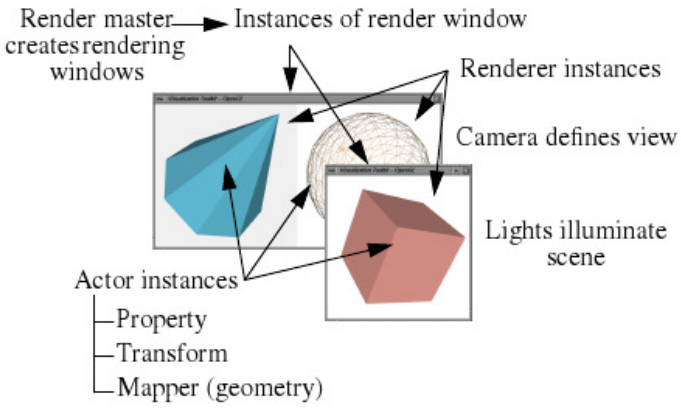

Figure 3: VTK graphic model[8]

The following are the basic objects in the model[9], each with a description of its functionalities.

1. Render Master - coordinates device-independent methods and creates rendering windows.
2. Render Window - manages a window on the display device. One or more renderers draw into a render window to generate a scene (i.e., final image).
3. Renderer - coordinates the rendering of lights, cameras, and actors.
4. Light - illuminates the actors in a scene.
5. Camera - defines the view position, focal point, and other camera characteristics.
6. Actor - an object drawn by a renderer in the scene. Actors are defined in terms of mapper, property, and transform objects.
7. Property - represents the rendered attributes of an actor including object colour, lighting (e.g., specular, ambient, diffuse), texture map, drawing style (e.g., wireframe or shaded); and shading style.
8. Mapper - represents the geometric definition of an actor and maps the object through a lookup table. More than one actor may refer to the same mapper.
9. Transform - an object that consists of a 4x4 transformation matrix and methods to modify the matrix. It specifies the position and orientation of actors, cameras, and lights.

In the FoB application, VTK will mainly be used for the visualization of the skeletal model. The model will move according to real time or recorded motion. Each bone is loaded into the application separately and added to a mapper. Then, actors referring each to a mapped bone is set to the right position. During the visualization, the angle and position of each actor and thus bone, will be updated according to the real time or recorded data of the corresponding sensors.

5 Medical Aspects

The current application we will be working on supports motion tracking for shoulder movements. Although it might be possible to track other parts of the body with the system, we will currently be focusing on the shoulder and its anatomy. To familiarize ourselves with the terms we have looked into the human anatomy briefly. Figure 4 displays the skeletal structure of a human body.

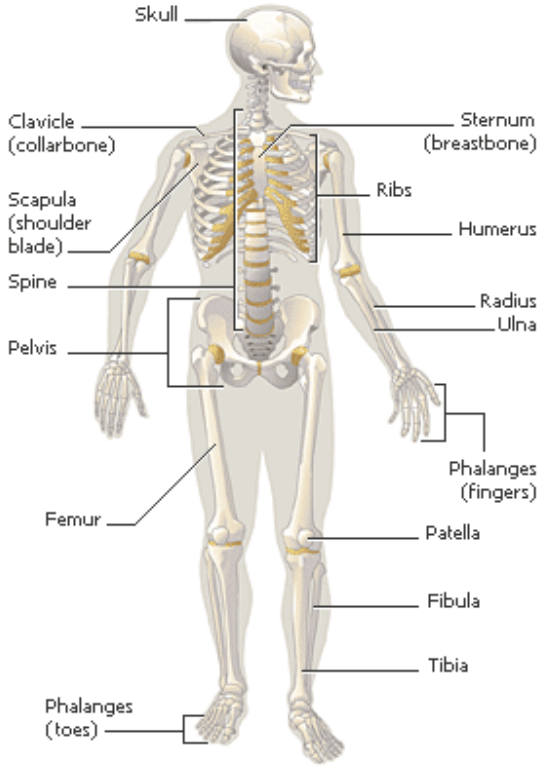

Figure 4: Human skeletal structure[10]

The human skeleton is a strong, flexible framework of 206 bones that supports the body and protects internal organs. The bones of the skeleton connect at joints to permit a wide range of body movements [10].

Through the tracking of this wide range of movements, joint and skeletal problems can be diagnosed and results of physical therapy can be detected. The three dimensional tracking application we will be working on can be a helpful tool for medical specialists. Studies have shown[11] similar tracking methods have proven successful in the diagnosis of medical problems such as frozen shoulders, a medical condition which causes pain and distinct movement patterns for the patient. Moreover, the system used in the study was also able to detect the results of clinical improvement.

6 References

[1] *Flock of Birds - Six Degrees of Freedom Measurement Device - Technical Description of DC Magnetic Trackers*
Ascension Technology Corporation, Burlington, VT
http://www.5dt.com/downloads/3rdparty/fobtechnicalspec.pdf
Retrieved on 06-10-08

[2] *OptotrakProSeries Image*
http://www.central-scanning.co.uk/images/OptotrakProSeries.jpg
Retrieved on 03-10-08

[3] *The Optotrak System*
http://p3.smpp.northwestern.edu/BMEC66/stroke/Optotrak.htm
Retrieved on 03-10-08

[4] *Programming Python: Object-Oriented Scripting (2001)*
Mark Lutz
http://books.google.nl/books?id=37_AJDlEytEC&printsec=frontcover&hl=en
Retrieved on 28-09-08

[5] *Pydev*
http://pydev.sourceforge.net/
Retrieved on 06-10-08

[6] *What is wxPython*
http://wxpython.org/what.php
Retrieved on 28-09-08

[7] *Numerical Python (2001)*
David Ascher, Paul F. Dubois, Konrad Hinsen, Jim Hugunin, Travis Oliphant
http://www-int.stsci.edu/~jinger/ad5/web/ad4/irafx/docs/numpy.pdf
Retrieved on 28-09-08

[8] *What Is VTK?*
http://www.vtk.org/what-is-vtk.php
Retrieved on 06-10-08

[9] *The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization*
William J. Schroeder, Kenneth M. Martin, William E. Lorensen, GE Corporate Research & Development
http://www.vtk.org/pdf/dioot.pdf
Retrieved on 28-09-08

[10] *MSN Encarta: Human Skeleton*
http://encarta.msn.com/media_701508645_761552814_-1_1/Human_Skeleton.html
Retrieved on 26-09-08

[11] *Measurement of three dimensional shoulder movement patterns with an electromagnetic tracking device in patients with a frozen shoulder (2002)*

H M Vermeulen, M Stokdijk, P H C Eilers, C G M Meskers, P M Rozing, T P M Vliet Vlieland
http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=1753981&blobtype=pdf
Retrieved on 25-09-08

Appendix B: Plan of Approach Document

Preface

This plan of approach describes the conditions and planning made in order to successfully implement our Bachelor of Science project “Flock of Birds” (FoB). The project is carried out on behalf of the Technical University of Delft (TU Delft), faculty of Computer Science, in cooperation with the Medical University Centre Leiden (LUMC), department of orthopaedics.

0 Management summary

The project is aimed to improve the already existing codebase of the Flock of Birds (FoB) application. The most important goals to achieve this are:

- Implement the link between the FoB hardware and software.
- Make the software more generic, less hardcoded.
- Implement a Centre of Rotation function.

The above goals are to be implemented within the timeframe set for this project (420 hours), which will be spread out over six months.

1 Introduction

The FoB application currently consists of a GUI for making measurements, and a 3D model that can visualize recorded measurements. The application is not yet ready for practical use, because the communication between the FoB hardware and software has not been implemented yet.
Our goal is to implement this link between the FoB hardware and software, so that the application is ready for use. We will also make the application more generic, to provide support for more flexible use of the application. This means that in addition to the shoulder measurements already possible with the application, the application will also be able to perform measurements on other parts of the human body. The third goal is the realization of a Centre of Rotation (CoR) function for the shoulder.
The three goals above are our primary goals. In order to successfully complete the project, they must be implemented and working. Progress on these goals will be discussed in weekly meetings with the project supervisors. There are also several

optional goals that can be implemented after the primary goals have been completed, these optional goals will be discussed later in this document.

The structure of this document is as follows:
In chapter 2, we will give a detailed definition of the project assignment. Then, in chapter 3, we describe our approach to complete the project. Chapter 4 discusses the project organisation and conditions to be realized by the project participants, the project provider and possible third parties. Chapter 5 contains the planning of our project, including the plan of activities, milestones plan and others. After that, we discuss the quality assurance and conditions set for the quality in Chapter 6. The last chapter concerns any further plans.

2 Project assignment

2.1 Project environment

The project environment is the movement laboratory at the LUMC. In this laboratory, the FoB system is used to perform measurements on patients. All of the tests will take place in this environment. The development of the code however, can be done at the practical halls at the TU Delft or at home. The primary reason this project is carried out is because of the need for a link between the existing FoB hardware and software. When this task has been completed, the new software can replace the existing command line based FoB application.

2.2 Project Objective

The new FoB application will be used to help patients with problems in the glenohumeral joint and other parts of the human body. The new application has to catch any possible mistakes made during measurements so that patients don't have to go through the whole measurement process again. The application should also be able to visualize recorded movements with a 3D model.

2.3 Assignment definition

Implement a link between the FoB hardware and software, re-implement the existing codebase in a more generic way, and adding functionality for the CoR.

2.4 Products and services to be delivered

A working software application for the FoB hardware which, next to the already existing GUI and visualization, includes the functionality to communicate with the FoB hardware and support for different kinds of measurements.

2.5 Demands and restrictions

The finished product should at least have the primary goals implemented. This means that it has to be able to communicate with the existing hardware, be capable of

performing measurements on body parts other than the shoulder and should have functionality built in for a CoR measurement.
There are also several optional goals, listed below:

- Possibility to disable the visualization model.
- Possibility to load in different models for visualization (especially for children).
- Implement more flexibility for the placement of bony landmarks.
- Implement a more variable approach for making the landmarks.
- Implement support for the OptoTrak.
- Add timestamps to measurements.
- Implement a reset option for the visualization model to put the model back in a default position.

These optional goals are not required for the finished product, but will be realized after the primary goals have been completed.

2.6 Crucial success factors

A crucial factor that the project participants can influence is the knowledge of the programming language Python and the Visualization Toolkit (VTK). Furthermore, knowledge and understanding of the existing codebase is also needed. It is therefore of great importance that the participants study these subjects in depth.

3 Approach

3.1 Phase 1: Orientation

In the first phase we will familiarize ourselves with Python, VTK, the FoB hardware and the skeletal anatomy. This preparation is described in the orientation report.

3.2 Phase 2: Design

In the design phase, we will first analyse the existing class diagrams of the FoB application, and then adjust them to reflect the current situation and to add new classes for our implementation. This phase has to be completed before beginning implementation to ensure that the implementation is done based on good design principles.

3.3 Phase 3: Implementation

Phase 3 is where we will actually implement our goals as described earlier in this document. The result of this phase should be a working FoB application with the desired functionality.

3.4 Phase 4: Testing

During this phase, we will have to test our software thoroughly at the movement laboratory at the LUMC to ensure that the application has no bugs and to check for possible other problems.

4 Project organisation and conditions

4.1 Project Organisation

The organisation of this project will be described using the OPAFIT model.

4.1.1 Organisation

- Dr. Ir. J. H. de Groot, Rehabilitation Medicine, LUMC
  Project provider
- Drs. F. Steenbrink, Orthopaedics, LUMC
  Project provider
- Dr. Ir. C. P. Botha, Medical Visualization, TU Delft
  Project supervisor
- Ir. P. R. Krekel, Medical Visualization, TU Delft & Orthopaedics, LUMC
  Project supervisor
- Peter Curet
  Project participant
- Mark Krijgsman
  Project participant

4.1.2 Personnel

The personnel using the product are also the project providers. There is no need for them to learn any new skills in order to use the newly developed system, since an older version of the FoB application is already in use and this system only simplifies use of that application.

4.1.3 Administrative procedures

The administrative procedures consist of a Plan of Approach, a technical report of the system and a final report that gives a detailed description of the finished product.

4.1.4 Financing

There is no financing needed for this project because all the required hardware is provided by the LUMC and TU Delft, and the software being used is open source.

4.1.5 Information

During the project, there will be a weekly meeting between the project participants and project supervisors. At these meetings, the progress of the project and possible questions will be discussed. Meetings at the LUMC will also be held, but this will mostly take place in the testing phase when the software needs to be tested with the hardware setup.

4.1.6 Technology

The hardware being used is the FoB system at the LUMC movement laboratory. This laboratory will also be our place of work during tests. The software we are using (Python, VTK and other plug-ins) are open source and thus freely available. During development of the software, our place of work will be the practical halls at the TU Delft.

4.2 Conditions for project participants

Conditions for the project participants are:

- The project will be completed within the scheduled 6 months.
- The implementation is based on good designing principles and is well documented.
- The project is concluded with a final report and a presentation of the finished product.

4.3 Conditions for project providers

Conditions for the project providers are:

- The project participants will be given the possibility to analyse and test on the FoB hardware.
- The existing code will be made available to the project participants for their use.
- Provide further information where needed and be available to answer any questions regarding the FoB system.

5 Planning

In the following chapter we will provide an estimate of the balance between our activities, time and resources spent while working on the project.

5.1 Norms and Assumptions

One of the primary assumptions we will make is that the hardware we will be using in the laboratory will not be readily available to us at every moment. We will be dependent on the schedule and availability of the medical specialists working in the laboratory, and our supervisors.
This might cause several delays throughout the process, as we are dependent on planned appointments. We should take this factor into consideration, particularly during the implementation phase and during the testing of our application.
Furthermore, we will take into assumption that whenever we are having difficulties during the development, our supervisors will provide help to us.
Another assumption made is that, while we will aim to follow the norm of pursuing the planning detailed in the next sections, it will most likely change during the project. It is to be expected that we will have to focus on unforeseen problems which may arise, resulting in the need for the schedule to be reorganized.

5.2 Plan of Activities

During the project we have fixed main activities planned and repeated activities which will be repeated periodically. We will elaborate on these activities in this section.

5.2.1 Main Activities

The main activities outline the process of development. The entire course of the project will run from begin September through the end of January spanning one semester of the year, as classified by the TU Delft.

- ***Orientation***
  Week 1 - 4
  15 September 2008 – 6 October 2008
- ***Design***
  Week 4 – 7
  6 October 2008 – 31 October 2008
- ***Implementation***
  Week 7 – 19
  31 October 2008 – 15 January 2009
- ***Testing***
  Week 16 – 20
  5 January 2009 – 23 January 2009
- ***End Report***
  Week 14 – 20
  15 December 2008 - 23 January 2009

In Figure 1 the process is summarized in a Gantt-chart.

ID	Task Name	Start	Finish	Duration
1	Orientation	9/15/2008	10/6/2008	3w 1d
2	Design	10/6/2008	10/31/2008	4w
3	Implementation	10/31/2008	1/15/2009	11w
4	Testing	1/5/2009	1/23/2009	3w
5	End Report	12/15/2008	1/23/2009	6w

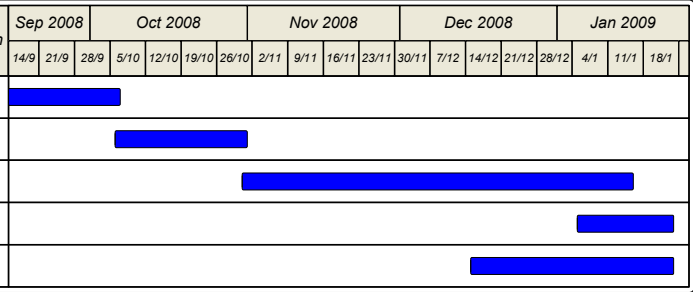

Figure 1: The planned development process displayed in a Gantt-chart

Each activity will be further elaborated in the following subsections.

Orientation

During the project we will be working with several different technologies. To familiarize ourselves with these technologies, it is important to conduct research on the relevant areas of computer and medical science. The results of this research will be described in the orientation report. Furthermore, to acquaint ourselves with the domain of the application, we will visit the movement laboratory at the LUMC. In addition, several meetings with our supervisors will help us learn about the assignment and its

development. Furthermore, analysis of the previous project is an important factor in the orientation phase. The source code will be examined carefully in order to fully comprehend the current application.

Design
Once the orientation process has been completed, the design of the FoB visualization application will be adjusted and new elements will be added. Seeing as there is already a design made for the application, our task will be to find faults in the design and try to improve them. Careful analysis of the primary goals assigned to us is needed, in order to incorporate them into the existing design.

Implementation
During the implementation phase, we will realize the design modifications and additions shaped in the previous design process. This phase is estimated to take the longest of all activities, as it is the most intensive process. This activity is usually combined with a portion of the testing phase.

Testing
Once the majority of the newly designed elements have been implemented into the FoB visualization software, it will be important to test the application. We will be running tests of the recorded data, which can be performed at home. The real-time movement visualization however, needs to be tested with the hardware in the LUMC movement lab. Therefore we will be working in the laboratory several times to assure the software is functioning correctly.
It is also important to note that during the process of the implementation, we will be testing thoroughly as well, as it is an imperative activity usually performed during programming.

End Report
Once the previous phases have been completed, an end report will be written, documenting the entire process and detailing the previously mentioned activities.

5.2.2 Repeated Activities

During the project, we have several activities which will be repeated periodically.

Meetings
It will be important throughout the project to talk to our supervisors, as their help and advice will help us steer the process into the desired direction. It is important to plan these meetings on a regular basis, generally every week.

Documenting
For the period of the project, it is important to document the process. This will be essential during the writing of the end report, as we will have much of it written by the end of the testing phase. This will ease and shorten the time spent on writing the report.

After each meeting we will write a short report detailing the important points. Furthermore, we have decided to document the amount of time of each session when working on the project. This will help us calculate and keep track of our efforts.

5.3 Milestones and Product Plan

The following are the important milestones and the estimated product plan of our project. These are the significant moments when we will be judged and graded based on the work we have been doing.

- **Orientation Report**
  6 October 2008
- **Design Document**
  31 October 2008
- **Implementation**
  - **Real-time link between hardware and software**
    25 November 2008
  - **Generic System**
    20 December 2008
  - **Centre of Rotation**
    15 January 2009
- **Testing**
  23 January 2009
- **End Report**
  23 January 2009
- **Presentation**
  End January 2009

5.4 Resource Plan

As mentioned earlier in this chapter, resources such as the tracking hardware in the laboratory of the LUMC will not be available to us at every moment. Appointments will have to be made in order to test our software with the hardware. This will mainly be during the implementation and testing phase.
Furthermore, transportation is an important resource needed in order to reach the lab. We will be making use of public transportation to travel between Leiden and Delft.
Other resources such as computer hardware are readily available to us, both at home and in the practical rooms of the faculty of Computer Science.
Since all the software we will be using to develop the FoB vizualizer is open source, we can download it through the internet.
Lastly, our available time is an important resource which we will need to plan and balance correctly in order to complete our desired goals.

5.5 Financial Plan

We will look briefly into the financial aspects of the project, as it is not a significant factor in the success of the development, mainly because all of the resources are freely available to us.
The FoB hardware is provided to us by the lab, computer hardware is available to us for free as well.
Furthermore, though we will be travelling between Delft and Leiden through public transport, both of us have student public transport card enabling us to travel without paying during weekdays when we will be working on the project.
And as mentioned earlier, the software used for the development is open source, and thus available freely.

6 Quality Assurance

In order to meet the quality standards expected by the LUMC movement laboratory personnel and our supervisors, we will look into the aspects of quality on which we will concentrate during the development of our product. Furthermore, we will take measures during the development to minimalize certain risks in order to meet the standards expected from the project.

6.1 Product Quality

The following requirements have been set concerning the quality of our product

- **Usability**

  An important aspect is to maintain and to improve on the usability of the current application. Efficiency of use and a small learning curve for novices are important aspects we will sustain.

- **Efficiency**

  The software needs to be efficient in order to provide its functionality. Mainly during the visualization of the movements, smooth and consistent animation is desired.

- **Reliability**

  As the software will be used in the medical domain, a large focus on the reliability is needed. As the patient's wellbeing is reliant on the correctness of the software, an emphasis will be made to maintain this aspect of product quality.

- **Allowances for maintainability and enhancement**

  As time passes and new tracking technology emerges, it is desirable to be able to connect these new technologies to the visualizer with minimal effort.
  While our project supervisors have already been working on this aspect of the software, we will continue the project making sure this requirement is maintained.

6.2 Process Quality

To ensure the entire process is worked through smoothly, we will have to ensure optimal teamwork. We will be using Subversion (SVN) in order to collaborate easily. Furthermore, we will maintain the proposed measures detailed in the following section during the entire development period.

6.3 Proposed Measures

To maintain the product and process quality the following measures will be taken:

- Documentation of code
- Documentation of revisions
- Periodic meetings with supervisors and project providers at the LUMC lab
- Frequent testing of our software during and after development

Appendix C: Requirements Analysis Document

1 Introduction

In this document we will discuss the requirements for the new Flock of Birds (FoB) application. The application will not be built from scratch, but will be further developed from an existing codebase, aimed at replacing the existing DOS FoB application. This project aims to improve the existing codebase and to make it ready for practical use by the Medical University Centre Leiden (LUMC).

2 Current situation

In the current situation an application has been developed, improving on the old DOS-based FoB application. This new application however, is not yet ready for practical use. Below, each of the four main tabs of the improved application are described first, followed by some general properties of the current system.

- For a measurement to be made, there are seven sensors that have to be attached to the patient. Three sensors per arm, namely on the lower arm, upper arm and acromion, and also one on the thorax. These sensors are visualized in a sensor panel and can be exchanged with each other if needed.
- When the sensors have been attached, the positions of the bony landmarks need to be calibrated, where bony landmarks are specific points on the bones that correspond with one or more of the attached sensors. This calibration is done with the use of a separate stylus sensor. Each bony landmark is measured five times by the stylus sensor, and with each measurement the stylus is tilted in a different angle. Then, these measurements are averaged to acquire a precise position. The GUI of this protocol has been designed in such a way that the user can see which measurements have to be made and whether they have been executed correctly. However, the actual functionality for making the measurements and checking for faulty measurements has not yet been implemented.
- A separate panel for the Glenohumeral Joint (GH-Joint) exists, which is meant to calculate the GH-Joints using either a regression or a movement method. Both of these methods are listed in this panel, but the functionality is yet to be implemented.
- The visualizer functionality has been partially implemented; a skeleton model is automatically loaded, and movement data to be visualized by the skeleton model can be imported.
- Many methods have been implemented based on hardcoded parameters, using hardcoded stylus sensor names and fixed lengths for bony landmark and sensor arrays.
- There is no link between the software and the hardware that is being used for the measurements.

3 The New System

3.1 Overview

We will now give an overview of the changes we will be making to create the new system. Each panel will be discussed, beginning with the Sensors panel.

- When starting up the application, the Sensors panel will automatically configure the current set of sensors two through nine. With the use of colours, feedback will be given concerning the position of the sensors.
  We have studied the possibility of having a separate advanced menu that is fully GUI based. This menu would allow the user to create, modify and delete sensors and corresponding bony landmarks. Bony landmarks can be appointed on a 3D-model. Furthermore, the rule base that checks for correct measurements can be adjusted in this menu as well.
  However, due to time constraints we have decided to implement a more hybrid approach where only a part of the menu is present in the GUI, and the rest is to be edited manually by an advanced user. This hybrid approach gives the user the ability to appoint and acquire bony landmarks coordinates on a 3D-model, but requires the user to manually edit the configuration files to add the name, description, rule, location and picture of a bony landmark.
- In panel two, the Bony Landmarks panel, the actual functionality of making and checking a measurement will be realised. The list of bony landmarks will also display bony landmarks which may have been created in the Sensors panel.
- In the GH-Joint panel, the regression and movement method will be selectable. The movement method requires the patient to make circular movements with his or her arms. Following these measurements, the centre of rotation can be estimated. The functionality for measuring these movements and to calculate the GH-Joint will be implemented as well.
- The Visualization panel will be extended to give support for different 3D-models to be loaded. There will also be an option to reset or disable the visualization model.
- The code will be adjusted to minimize hardcoded parameters and to support more variable options, such as extra flexibility concerning the placement and creation of bony landmarks. Timestamps will also be added to measurements.
- Specific drivers for communication between software and hardware of the FoB and OptoTrak system will be implemented.

3.2 Functional requirements

There are two actors who will make use of the system. Both of these actors are researchers, where the first actor acts as a standard user and the other as an advanced user.
The standard actor will be able to perform the following actions:

- Attach sensors to the patient's body which will be checked for correct placement.
- Perform bony landmark measurements. Each bony landmark will be measured five times and coloured to indicate whether it is measured correct or not.
- Calculation of the GH-Joint(s), using either the regression or movement method. The regression method uses existing measurement for calculation, and the movement method prompts a menu guiding the user in making the new measurements.
- Record and save a new measurement in real time using the 3D-model in the Visualization panel.
- Playback recorded measurements in real time using the 3D-model in the Visualization panel.

The advanced actor will be able to perform all the actions of the standard actor, plus a set of additional actions listed below:

- Configure locations of the sensors on the body manually in the configuration file.
- Add, modify and delete new bony landmarks. When adding a bony landmark, the actor can select the location in a 3D-model, and manually edit the name, description, rule and picture in the configuration file.

3.3 Non-functional requirements

3.3.1 User interface and human actors

The user interface has to be intuitive and easy to use. The user has to be able to reach his or her goal in as few steps as possible. It should always be clear what the progress of the user is during measurements of the bony landmarks. Each bony landmark measurement is coloured to indicate whether a measurement is correct or not. In the Sensors panel, colours will also indicate whether the sensors have been placed correctly.

3.3.2 Documents

The following two documents are required for the system; the user manual, which can guide the user trough the application, and a documentation of all the functionality of the source code. The latter document can be consulted when there is need for maintenance or extensions to be made to the existing program. This is an important document, considering the fact that the software will be released as open source.

3.3.3 Hardware considerations

The application might be used for different sorts of hardware, namely the FoB system and OptoTrak system. This requires the software to be able to support both.

3.3.4 Performance characteristics

The system should be able to produce a fluid visualization of the model, both for recorded and real-time data.

3.3.5 Error handling and extreme conditions

Errors need to be handled with the use of user-friendly GUI messages, with information on the error and appropriate steps to be taken to correct the problem. Errors of any kind may not crash the system. Extensive error checking needs to be implemented especially in situations where the user works with manually edited configuration files. Consistency should be maintained at any time. This means that users cannot navigate to the next menu when new sensors and bony landmarks have not yet been configured or when measurements have not been fully completed.

3.3.6 System modifications

The system has to have a design that is aimed on extensibility, so that extra functionality can easily be added to the system at a later stage if needed.

3.3.7 Physical environment

Since the system is influenced by electromagnetic fields, the environment needs to be free of distortion of any such kind. If the user does not take this aspect into account, measurements can be negatively influenced. Before using the system in a new environment, it needs to be calibrated to account for any present electromagnetic fields.

3.4 Constraints

The application will be implemented using the programming language Python. Editing of the code will be done using ConTEXT. The following libraries and plug-ins are used during development:

- Visualization Toolkit (VTK), for the development of 3D-models.
- wxPython, for GUI design
- NumPy, to extend the mathematical functionality of Python
- Subversion (SVN), for version control and file sharing
- Pydoc, to generate documentation for the system.

4 System models

4.1 Use cases

We will now distinguish the several use cases for the application. These uses cases are derived from the actions the actors can take as described in section 3.2:

1. Attaching a sensor to the patient's body.
2. Adding, modifying and deleting new bony landmarks and sensors.
3. Performing bony landmark measurements.
4. Calculation of the GH-Joint(s), using the regression and movement method.
5. Record and save a new measurement.
6. Playback recorded measurements.

Use case: attaching a sensor to the patient's body
Actor(s): standard actor/advanced actor
Precondition(s): the system has just been started, the actor is in the starting Sensor panel and the patient is seated in the FoB unit.

Actor actions	System responses
1 Attach the sensors to the patient and click on the "Next" button.	2a Verifies that all the sensors are placed correctly and proceeds to the Bony Landmarks panel.
	2b Notifies the user that a sensor has been placed wrong and marks the incorrectly placed sensors orange on a 3D-model.
3 Checks all the sensors and corrects the sensors that have been placed in a wrong position, then clicks the "Next" button.	4 Goes to step 2a.

Use case: adding, modifying and deleting a new bony landmark
Actor(s): advanced actor
Precondition(s): user has not yet started a measurement or recording session
Add:

Actor actions	System responses
1 Selects the "Add bony landmarks" option from the "Edit -> Add bony landmarks" drop down menu.	2 Creates a dialog showing a 3D-model and a list of custom made bony landmarks.
3 Selects the desired position for the new bony landmark on the 3D-model	4 Displays the point clicked with a marker on the model
5 Presses the "Add" button.	6 Adds the new bony landmark's 3D coordinates to the list.
7 Clicks on another point on the 3D-model	8 Displays the marker on the new point

9 Presses the “Finish” button.	10 Notifies the user that the 3D coordinates for the new bony landmark have been copied to the clipboard and instructs the user to manually edit the configuration file, where he/she has to add the name, description, rule and picture location.
11 Presses “OK”.	12 Closes the dialog.

Modify/Delete:

Actor actions	System responses
1 Opens the configuration file and modifies or deletes the lines relevant for the bony landmark(s) to be adjusted.	3 Remains idle until the system has been restarted.
2 Saves the edited configuration file.	
4 Restarts the system.	5 Reads the new configuration file.

Use case: measuring the bony landmarks
Actor(s): standard actor/advanced actor
Precondition(s): the actor has attached the sensors correctly to the patient and is in the Bony Landmarks panel.

Actor actions	System responses
1 Selects the first bony landmark in the shown list.	2 Shows the corresponding bony landmark picture.
3 Places the stylus sensor on the patient’s body and presses the “Pedal” button.	4 Colours the first measurement, where green stands for OK and orange for a faulty measurement.
5 Repeats step 3 up to five times.	6 Repeats step 4 up to five times. After the fifth time, moves on to the next bony landmark on the list automatically.
7 Repeats the above steps until all bony landmarks are measured.	8 Repeats the above steps until all bony landmarks are measured.
9 Presses the “Next” button.	10a Indicates that one or more bony landmarks have not been measured correctly.
	10b Proceeds to the GH-Joint panel.

Use case: calculating the GH-Joint(s), using the regression and movement method
Actor(s): standard actor/advanced actor
Precondition(s): The actor has attached all sensors and measured all bony landmarks successfully. The actor is now in the GH-Joint panel.

Regression method:

Actor actions	System responses
Selects the “Regression” bullet and then presses the “Next” button.	Calculates the GH-Joint and proceeds to the Visualization panel.

Movement method:

Actor actions	System responses
1 Selects the “Movement” bullet.	2 Enables the menu attached to “Movement”.
3 Selects the “Start/Stop Left GH-Joint” button.	4 Starts measurement.
5 Instructs the patient to make the required movements and then presses the “Start/Stop Stop Left GH-Joint” button again.	6 Stops measuring and calculates the left GH-Joint immediately. After calculation the button is coloured green to indicate the calculation was successful.
7 Selects the “Start/Stop Right GH-Joint” button.	8 Starts measurement.
9 Instructs the patient to make the required movements and then presses the “Start/Stop Stop Right GH-Joint” button again.	10 Stops measuring and calculates the right GH-Joint immediately. After calculation the button is coloured green.
11 Presses the “Next” button.	12 Proceeds to the Visualization panel.

Use case: record and save a new measurement
Actor(s): standard actor/advanced actor
Precondition(s): The actor has attached all sensors and measured all bony landmarks successfully.

Actor actions	System responses
1 Presses the “Record” button.	2 Starts recording all movements made by the patient.
3 Presses the “Stop” button.	4 Creates a dialog asking where to save the recording.
5 Selects the location and presses OK.	6 Saves the recording to disk and adds the recording to the list of “Available recordings” for playback.

Use case: playback of a recorded measurement
Actor(s): standard actor/advanced actor
Precondition(s): The actor either has attached all sensors and measured all bony landmarks, or has just started the program.

Actor actions	System responses
1 Presses the “Import” button.	2 Creates a dialog asking where to load the recording from.
3 Selects the location and presses OK.	4 Loads the selected recording into the “Available recordings” list.
5 Selects the recording in the list and presses the “Play/Pause” button.	6 Visualizes the recording real-time.
7 Presses the “Play/Pause” button.	8 Stops visualizing.

4.2 Scenarios

We will now use the above defined use cases to describe different scenarios.

Scenario: Adding a new bony landmark
Actor(s): John, standard actor and Peter, patient

Actor actions	System responses
1 John selects the “Add bony landmarks” option from the “Edit -> Add bony landmarks” drop down menu.	2 Creates a dialog showing a 3D-model and a list of custom made bony landmarks.
3 John selects the desired position for the new bony landmark on the 3D-model [coordinates (246.6, 198.2, -1087.9)] and presses the “Add” button.	4 Adds the new bony landmark’s 3D coordinates to the list.
5 John presses the “Finish” button.	6 Notifies John that the 3D coordinates for the new bony landmark have been copied to the clipboard and instructs him to manually edit the configuration file, where he has to add the name, description, rule and picture location.
7 John presses “OK”.	8 Closes the dialog.
9 John opens the configuration file and scrolls to the [BonyLandmarks] section. There, he adds the line “25: Processus Xiphoideus” at the end of the list.	13 Remains idle until the system has been restarted.
10 Then he scrolls to the [BL Descriptions] section where he adds the line “25: The most caudal point of the sternum” at the end of the list.	

11 Next, he scrolls to the [BLRulebase] section where he adds the line “18:x 13 > 23” at the end of the list. He then saves and closes the configuration file.	
12 John also wants to add a picture for the new bony landmark, so he browses to the \FoB\Images\BonyLandmarkImages\ folder and pastes the new image named BonyLandmark 25.png	
14 Restarts the system.	15 Reads the new configuration file.

Scenario: Performing bony landmark measurements
Actor(s): John, standard actor and Peter, patient

Actor actions	System responses
1 John selects the “Processus Xiphoideus” from the bony landmark list.	2 Shows the corresponding bony landmark picture.
3 John places the stylus sensor on the Peter’s body and presses the “Pedal” button.	4 Colours the first measurement green.
5 Repeats step 3 up to five times.	6 Repeats step 4 up to five times. After the fifth time, moves on to the next bony landmark on the list automatically.
7 Repeats the above steps until all bony landmarks are measured.	8 Repeats the above steps until all bony landmarks are measured.
9 John presses the “Next” button.	10 Indicates that the Left Trigonum Spinae has not been measured correctly.
11 John selects the Left Trigonum Spinae that has been coloured orange to indicate a faulty measurement.	13 Resets all measurements for the selected Left Trigonum Spinae.
12 John presses the “Reset selected” button.	
14 Repeats step 3 up to five times again for a new measurement.	15 Repeats step 4 up to five times again.
16 John presses the “Next” button.	17 Proceeds to the GH-Joint panel.

Scenario: Calculation of the GH-Joint(s), using the regression and movement method
Actor(s): John, standard actor and Peter, patient

Actor actions	System responses
1 John selects the “Movement” bullet.	2 Highlights the menu attached to “Movement”.
3 John presses the “Start/Stop Left GH-	4 Starts measurement.

Joint" button.	
5 John instructs Peter to make the required movements and then presses the "Start/Stop Left GH-Joint" button again.	6 Stops measuring and calculates the left GH-Joint immediately. After calculation the button is coloured green.
7 John presses the "Start/Stop Right GH-Joint" button.	8 Starts measurement.
9 Instructs Peter to make the required movements and then presses the "Start/Stop Right GH-Joint" button again.	10 Stops measuring and calculates the right GH-Joint immediately. After calculation the button is coloured green.
11 John realizes a measurement for the right GH-Joint is not needed so he presses the "Reset" button.	12 Asks the user whether to delete the left, right, or both GH-Joint measurements.
13 John confirms deletion of the right GH-Joint and then presses "Next".	14 Alerts the user that a measurement for the right GH-Joint has not been made and ask for confirmation to continue (Yes/No).
15 John presses "Yes".	16 Proceeds to the Visualization panel.

Scenario: playback of a recorded measurement
Actor(s): John, standard actor

Actor actions	System responses
1 John selects the "Import" option from "File -> Import".	2 Creates a dialog asking where to load the recording from.
3 John browses to C:\FoB\data\P24MOV.DAT and presses "OK".	4 Loads the selected recording into the "Available recordings" list.
5 John selects P24MOV.DAT in the list and presses the "Play/Pause" button.	6 Visualizes the recording real-time.
7 John presses the "Play/Pause" button.	8 Stops visualizing.

Appendix D: Object Design Document

1 Introduction

In this document we will describe the design of the existing Flock of Birds (FoB) application. Because the source code will be released as open source, it is important to document the existing system design in English to increase its accessibility. Furthermore, changes and design decisions that we have made concerning the existing design will be further elaborated on.
In the next chapter we will discuss the main design elements of the application. We will also elaborate on the design decisions we have made to further improve on the application. In the third chapter we will discuss the main packages in which the software can be split up in. In the fourth chapter we will talk about the state machine architecture of the software, and in the last chapter we will elaborate on the complete code architecture by updating the class diagram and discussing class descriptions.

2 Overview

In this section we will give a summarized description of the main components and ideas of the application. Furthermore, we will discuss our main decisions concerning changes we have made to the previous design.

2.1 Main Design Description

As we are working on an existing application, our main objective is to maintain the existing ideas of the previous design, and further improve on them where necessary. Because changes have been made in the code, it is important to document the progress, analyse the changes, and to update the corresponding design diagrams.

The main design can be split up into four packages, each having a distinct function. The main package, the reader, the GUI and the visualizer. Furthermore, a state machine design pattern is utilised. This state machine pattern implies that the program can be in one of the defined states, each with different functionality. Each state corresponds with a panel that the GUI can display.

2.2 Design Decisions

One of our main objectives is making the code more generic. This decision implies removing any hardcoded elements existent in the code. Resulting from this, more advanced users will be able to modify and add configurations through the configuration.ini file. The application needs to be able to accept and process these changes.

One of the dilemmas we have faced is the possibility of having an option for advanced functionality for the users. Although we have developed this idea in our requirement analysis document, we have decided on excluding it from the design, leaving the advanced functionality as an optional goal. Instead we will focus our efforts on the functionality mentioned earlier of modifying the configuration file, a non-GUI method for modifying configurations for advanced users. This exclusion will benefit the visibility of designed diagrams, preventing unnecessary clutter.

Furthermore, we want to further the possibility of connecting different hardware such as OptoTrak to the application. This will be done by allowing separate driver classes, each for the specific hardware, to be connected to the software. An interface hardware driver will be added to the design, allowing the specific hardware driver such as the *FoBDriver* to inherit basic functions. These primary functions will allow the main driver to retrieve sensor information from different hardware drivers in a consistent way. The backend however, will be a hardware specific implementation of the actual link between the hardware and software. Currently, hardware specific code is used in the main driver. Migrating misplaced GH estimation methods from the driver to the

FoBDriver will be essential. Furthermore many mathematical Matlab methods in *Driver* are unused and can be moved to the driver unused function file.
Throughout the application several class, variable and method names possibly causing misunderstandings have been changed, improving readability of the code.
Furthermore, a consistent checking method will be added for each state class assuring the correctness of each step made by the user.

3 Packages

The design of the software can be looked at as consisting of several separated packages. Each package consists of classes serving one function as a whole. Figure 1 displays a diagram containing the main idea of the packages.

3.1 Main

The main package contains the classes which form the core of the program, and control the entire process. This is done through a state machine design pattern described in the next section. Each state corresponds with a GUI package panel classes. This allows the graphical elements to be separated from the functional elements.

To improve on the clarity of this package design, the state entities *RecordingState* and *PlaybackState* in the main package have been displayed as one entity inside the *VisualizationState*. The *RecordingState* and *PlaybackState* were previously named *RealtimeState* and *RecordingState* in consecutive order. During the process of requirement analysis and designing, the conclusion was made that these names were inappropriate, as they caused several misunderstandings. The *RealtimeState* has now been renamed to *RecordingState*, as this is were the actual process of recording takes place. The *RecordingState* has been renamed to *PlaybackState* as the playback of movements takes place in that state. These changes will be carried out in the corresponding GUI classes as well.

Classes of the Main package communicate with both the GUI package and the Reader package.

3.2 Reader

The reader classes handle the input and processing of information. The *PreferencesFileReader* and the *ConfigFileReader* handle data concerning the settings of the application.

In the previous version of the application, functionality existed for opening and saving of previous measurement and recording sessions. These functionalities have been taken out of the current design, as they were only necessary for the simulation of sessions in the previous version of the application. As we are only keeping the functionality of importing previously recorded movements, the *FileLoader* class will be replaced by a

RecordingsManager class, which will only contain functionality for importing and exporting of movement data.

The driver handles the communication with the hardware and processing of the measurement data. It does this through the hardware specific drivers, for example, the *FoBDriver*. This driver will handle communication with the actual FoB hardware. We will be using a *FakeFob* class to simulate the hardware. When using other hardware such as the OptoTrak, an OptoTrak driver can be made and the *Driver* class will have to be adjusted to use it as the hardware driver. In the new design, each specific driver is implemented using an Interface class called *HardwareDriver*, to make sure all specific drivers will maintain a consistent implementation pattern with the same basic functionality.

3.3 GUI

This package handles the graphical elements of the program. Each screen of the program has a separate class. Several dialog menu's can also be found in the GUI package.

In previous design diagrams the GUI was displayed as being connected to the reader package. We believe however, that the GUI package needs to be separated and have the main package handle all communication with the reader package.

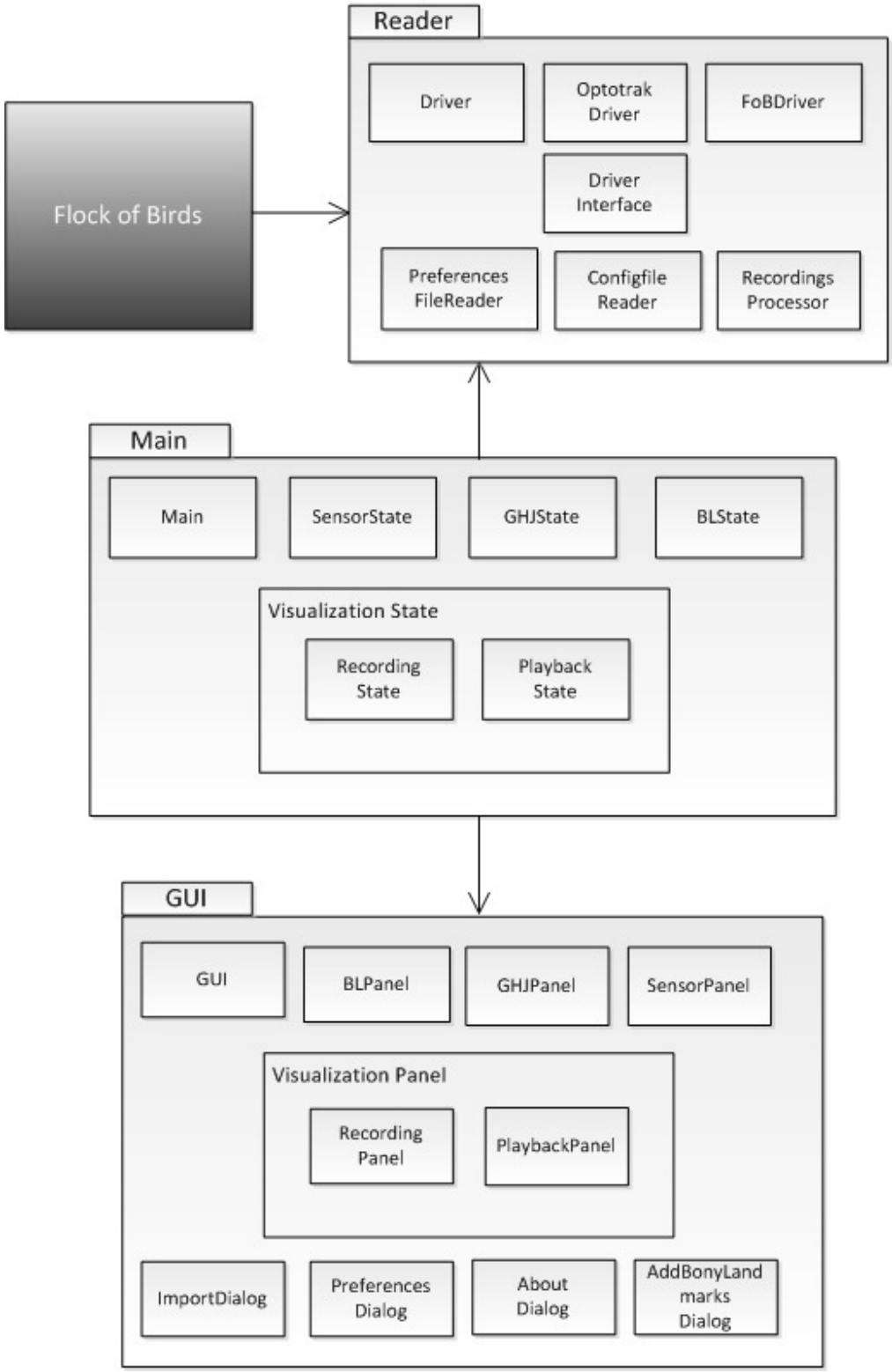

Fig. 1 A package diagram of the FoB visualizer

4 State Machine

One of the main design concepts prevailing throughout the code architecture is that of the state machine. We will maintain this design philosophy and apply it when further developing the software. Figure 2 displays the state machine design pattern of our application

The application is designed in such a way that the program is constantly in one of four states. The application allows going forward through the states whenever the necessary parameters are correct. Going forward through this path can only be done one state at a time. When residing in one of the later states however, the program can go back to any of its earlier states. Furthermore, each state has a separate corresponding GUI panel. The Visualization state has two sub-states not displayed in figure 2, namely the recording state and the playback state.

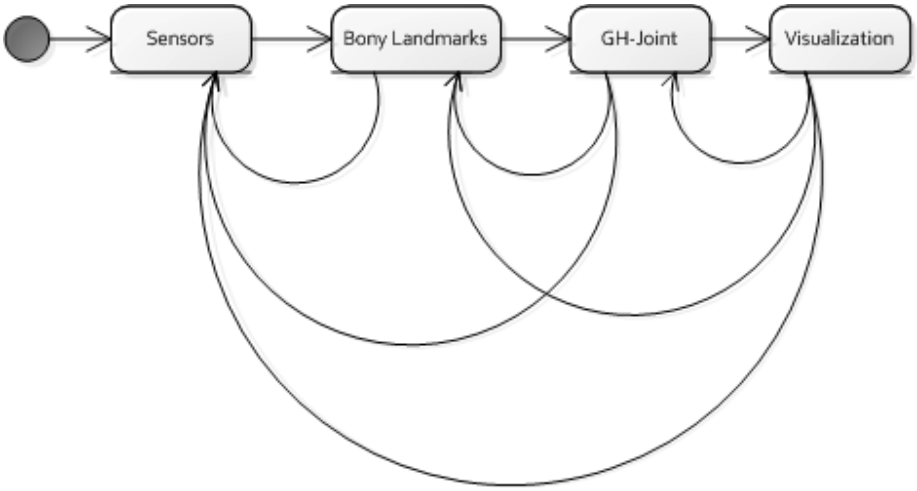

Fig. 2 A State machine diagram of the FoB visualizer

5 Classes

In the following section we will elaborate on the the design using class diagrams.
In figure 3 a summarized version of the class diagram is shown. Here the associations are clarified. For this version of the class diagram we included the *HardwareDriver* interface and let the *OptotrakDriver* and *FoBDriver* inherit from it.
Furthermore, the set of visualization panels and states have been reorganized.
In the previous version, the design showed the visualization sub-panels and states as only being accessible through the *VisualizationState* whereas in the updated diagram the sub-panels are connected to the *VisualizationPanel*, and the sub-states are connected to the *VisualizationState*. This was done to further consistent separation of the GUI classes from the State classes.

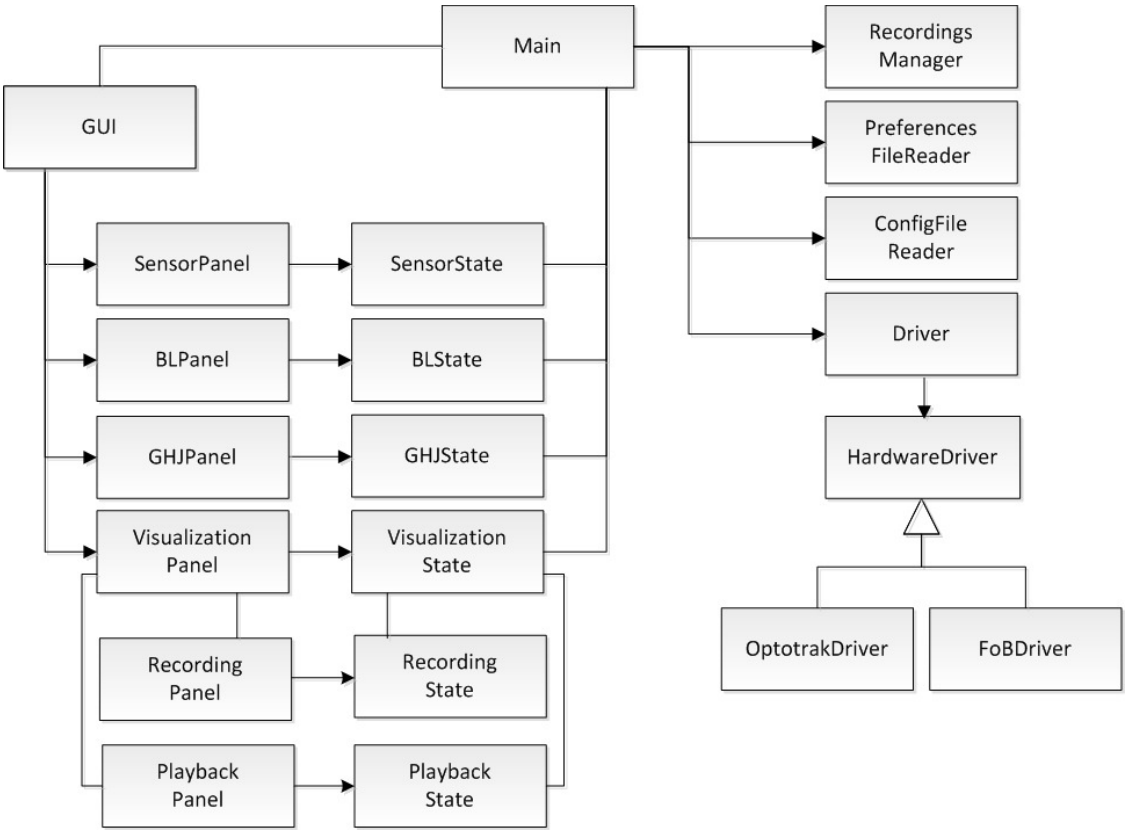

Fig. 3 A diagram of the important classes